Introduction to mathematical logic

This script accompanies PMATH 330 taught in Fall semester 2019

Dino Rossegger

Compiled: December 10, 2019

Preface

This book are course notes accompanying the course PMATH 330: Introduction to mathematical logic taught by the author at the University of Waterloo in the Fall semester 2019. Much of these notes is based on the material provided by Barbara Csima for the online version of this course.

At the current state the notes are not finished and will grow as we progress through the semester.

1 Introduction

As the main object of the title of this book is "Mathematical logic", we should start by saying what mathematical logic is. Doing this we might as well look what one of the greats of the field has to say about it:

> Logic is the study of reasoning; and mathematical logic is the study of reasoning done by mathematicians. — Shoenfield, Mathematical Logic

One feature of the field of mathematics that makes it a perfect candidate to study using the tools that formal logic gives us is that it is not an empirical science. Indeed, observations barely play a role in the everyday life of a mathematician. Given some statements the mathematicians wants to obtain his results by deriving them from these statements. So, what does it mean to "derive" a statement A from other statements B. Well, in the case of a mathematician it means that the mathematician writes down a proof that A follows from B. So in order to study "the reasoning done by mathematicians" we have to make precise what the verbs in the above sentences really mean. We have to give formal definitions of what a proof is, what it means to derive A from B and when A follows from B.

The goal of this book is to develop the formal languages needed to formulate the above and to equip the reader with the tools to study the "reasoning done by mathematicians".

Contents

2 Propositional logic

Consider the following argument which we will refer to several times in this chapter.

> Whenever it rains, the grass is wet.
> The grass is dry.
> Therefore, it is not raining.

We want to study reasoning, so we are interested in the structure of this argument. Why can we conclude that "it is not raining" given the arguments "the grass is wet" and "it does not rain". In order to study this question we want to formalize it in a formal language of our choice.

One way to do this is the following:

1. Let A stand for "it rains",
2. B stand for "grass is wet",
3. $\neg$ stand for "it is not the case that",
4. $\rightarrow$ stand for "implies that".

Then our argument would read as:

1.	$A \rightarrow B$	Assumption
2.	$\neg B$	Assumption
3.	$\neg A$	Conclusion

Notice how the choice of symbols A and B was somehow arbitrary. We thus removed the meaning of the arguments and reduced it to a form which lets us identify the structure of the argument better.

To further illustrate this consider the following exercise.

Exercise 2.0.1. Formalize the following arguments using the notation from above:

1. If the plum is green, then it is not ripe. The plum is green. Hence, it is not ripe.

2. Whenever the liar speaks, he lies. The liar does not speak. Therefore he does not lie.

2.1 Syntax of propositional logic

In order to study simple arguments as the ones presented in the introduction of this chapter we have to set up a framework. We therefore now define the *language of propositional logic*.

Building of a formal language like propositional logic is not so different from building a language like English. We start by defining its *syntax*, the set of rules that define the structure of a sentence (in the case of propositional logic formulas). In the next section we define a semantics for propositional logic, i.e., we give the formulas meaning.

In order to define the syntax of propositional logic we first give its basic building blocks, symbols.

Definition 2.1.1. The following are *symbols* of propositional logic.

- *Propositional variables* (Upper case letters with numerical indices):

$$\{A, B, C, \dots, A_0, A_1, \dots, B_0, B_1, \dots\}$$

- *Connectives*: $\neg, \rightarrow$

- *Brackets*: $(,)$

If we again want to draw a comparison to English, then the equivalent to propositional variables would be words, and the equivalent to connectives and brackets would be punctuation and spaces.

Not every sequence of words is an English sentence. Indeed, there are strict rules what a sentence in the English language is – the syntax. We can now define the syntax for propositional logic allowing us to build *propositional formulas*.

Definition 2.1.2. Propositional *formulas* are inductively defined as follows

1. All propositional variables are formulas.
2. If φ is a formula, then so is $\neg\varphi$,
3. If φ and ψ are formulas, then so is $(\varphi \to \psi)$.

We can now check whether certain strings are formulas. A proof that a certain string is a formula would look as follows.

Example 2.1.1. *The string* $\neg((P \to Q) \to R)$ *is a propositional formula.*

Proof.

1.	P, Q, R are propositional formulas	by 1 of def. 2.1.2
2.	$(P \to Q)$ is a p. formula	by 3 of def. 2.1.2 and 1
3.	$((P \to Q) \to R)$ is a p. formula	by 3 of def. 2.1.2, 1, and 2
4.	$\neg((P \to Q) \to R)$ is a p. formula	by 2 of def. 2.1.2 and 3

□

Exercise 2.1.1. Show that the following are propositional formulas.

1. $((P \to P) \to \neg P)$
2. $(((P \to Q) \to (P \to R)) \to (Q \to R))$

Note the inductive nature of Definition 2.1.2. We first define a base case, all the propositional variables. We then specify how to, given two formulas, build more complicated formulas. The nice thing about inductive definitions is that properties about these definitions can often be proved using, as the name suggests, induction. One example is the following.

Proposition 2.1.2. *All propositional formulas have the same number of left brackets as right brackets.*

Proof. Let φ be a propositional formula, l_φ be the number of left brackets in φ, and r_φ be the number of right brackets in φ. Our goal is to show that $l_\varphi = r_\varphi$. We show this by induction.

Base case: $\varphi = P$ for some propositional variable P. Then $l_\varphi = r_\varphi = 0$.

Induction step: We have two cases.

Case 1: $\varphi = \neg\psi$ for a propositional formal ψ with $l_\psi = r_\psi$. Then

$$l_\varphi = l_{\neg\psi} = l_\psi = r_\psi = r_{\neg\psi} = r_\varphi.$$

Case 2: $\varphi = (\psi \to \theta)$ for propositional formulas ψ, θ with $l_\psi = r_\psi$ and $l_\theta = r_\theta$. Then

$$l_\varphi = l_{(\psi\to\theta)} = 1 + l_\psi + l_\theta = 1 + r_\psi + r_\theta = r_{(\psi\to\theta)} = r_\varphi.$$

□

2.1.1 Subformulas

Notice how by Definition 2.1.2 a propositional formula is either a propositional variable or built from other propositional formulas. Formulas which are used in the building of a formula φ are called subformulas of φ. More formally:

Definition 2.1.3. Given a propositional formula φ we define the set $sub(\varphi)$ as follows.

1. If $\varphi = P$ for a propositional variable P, then $sub(\varphi) = \{\varphi\}$.
2. If $\varphi = \neg\psi$ for a propositional formula ψ, then $sub(\varphi) = \{\varphi\} \cup sub(\psi)$.
3. If $\varphi = (\psi \to \theta)$ for propositional formulas ψ and θ, then $sub(\varphi) = \{\varphi\} \cup sub(\psi) \cup sub(\theta)$.

If $\psi \in sub(\varphi)$, then ψ is a *subformula of* φ.

Example 2.1.3. *Calculate* $sub(\varphi)$ *for* $\varphi = \neg((P \to Q) \to R)$.

Solution.

$$\begin{aligned} sub(\varphi) &= \{\neg((P \to Q) \to R)\} \cup sub(((P \to Q) \to R)) \\ &= \{\neg((P \to Q) \to R), ((P \to Q) \to R))\} \cup sub((P \to Q)) \cup sub(R) \\ &= \{\neg((P \to Q) \to R), ((P \to Q) \to R)), (P \to Q), R\} \cup sub(P) \cup sub(Q) \\ &= \{\neg((P \to Q) \to R), ((P \to Q) \to R)), (P \to Q), R, P, Q\} \end{aligned}$$

◀

Exercise 2.1.2. to be added after assignment

2.2 Semantics of propositional logic

So far we know when a string is a formula, however we have not defined the meaning of formulas. In other words, we have not yet developed the semantics of propositional logic. Similar to the syntax, the definition of semantics of first order logic will again be inductive starting with propositions.

Definition 2.2.1. A *truth assignment* is a function e assigning to each propositional variable the value **T** (true) or **F** (false), i.e.,

$$e : PV \to \{\mathbf{T}, \mathbf{F}\}$$

Truth assignment provide the base case of our definition, assigning meaning to propositional values. In order to obtain semantics for all propositional formulas we have to define the semantics for connectives.

Definition 2.2.2. Given a truth assignment e we define a function $\hat{e} : PF \to \{\mathbf{T}, \mathbf{F}\}$ as follows.

1. If $\varphi = P$, a propositional variable, then $\hat{e}(\varphi) = e(P)$,

2. if $\varphi = \neg\psi$, for $\psi \in PF$, then

$$\hat{e}(\varphi) = \begin{cases} \mathbf{T} & \text{if } \hat{e}(\psi) = \mathbf{F} \\ \mathbf{F} & \text{if } \hat{e}(\psi) = \mathbf{T} \end{cases},$$

3. and if $\varphi = (\psi \to \theta)$ for $\psi, \theta \in PF$, then

$$\hat{e}(\varphi) = \begin{cases} \mathbf{T} & \text{if } \hat{e}(\psi) = \mathbf{F} \\ \mathbf{T} & \text{if } \hat{e}(\psi) = \mathbf{T} \ \& \ \hat{e}(\theta) = \mathbf{T} \\ \mathbf{F} & \text{if } \hat{e}(\psi) = \mathbf{T} \ \& \ \hat{e}(\theta) = \mathbf{F} \end{cases}.$$

We will usually write e instead of $\hat{e}$.

The definition of our semantics might seem arbitrary at first. Notice, however, how it mirrors our approach at the beginning of this chapter when we formalized the argument "Whenever it rains, the grass is wet. The grass is dry. Therefore, it is not raining". There, we said that $\neg$ stands for "it is not the case that" and $\to$ stands for "implies that". This is reflected in Definition 2.2.2.

We can now evaluate formulas in propositional logic.

Example 2.2.1. *Check whether* $(\neg(P \to Q) \to R)$ *is true under the assignment* $e : P \mapsto \mathbf{T}, Q \mapsto \mathbf{F}, R \mapsto \mathbf{F}$.

Solution. We calculate $\hat{e}((\neg(P \to Q) \to R))$ inductively using its subformulas $sub((\neg(P \to Q) \to R)$.

1. For the propositional variables the definition of $\hat{e}$ trivially is $\hat{e}(Q) = \hat{e}(R) = \mathbf{F}$ and $\hat{e}(P) = \mathbf{T}$,
2. $\hat{e}((P \to Q)) = \mathbf{F}$,
3. $\hat{e}(\neg(P \to Q)) = \mathbf{T}$,
4. $\hat{e}((\neg(P \to Q) \to R)) = \mathbf{F}$.

◀

If e is a truth assignment so that $e(\varphi) = \mathbf{T}$ we colloquially say that e *satisfies* φ.

2.2.1 Truth tables

It is useful to visualize the definition of $\hat{e}$ using *truth tables*. A truth table of a formula φ is a table which has as its columns the subformulas of φ and as its rows all the truth assignments. For example, the truth tables for our connectives are:

φ	$\neg\varphi$
T	**F**
F	**T**

φ	ψ	$(\varphi \to \psi)$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

We can write down truth tables for any propositional formula and can use these to evaluate the truth of a propositional formula under an assignment e by looking at the corresponding row.

Example 2.2.2. *A truth table for* $(\neg(P \to Q) \to R)$ *with the row representing* $e : P \mapsto \mathbf{T}, Q \mapsto \mathbf{F}, R \mapsto \mathbf{F}$ *highlighted.*

P	Q	R	$(P \to Q)$	$\neg(P \to Q)$	$(\neg(P \to Q) \to R)$
T	**T**	**F**	**T**	**F**	**T**
T	**F**	**T**	**F**	**T**	**T**
T	**T**	**T**	**T**	**F**	**T**
T	**F**	**F**	**F**	**T**	**F**
F	**T**	**F**	**T**	**F**	**T**
F	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**T**	**F**	**F**
F	**F**	**F**	**T**	**F**	**T**

Exercise 2.2.1. Calculate the truth tables for

1. $((\neg P \to Q) \to R)$,
2. $((P \to Q) \to (P \to Q))$,
3. $\neg(\neg P \to Q)$.

2.2.2 Validity, Satisfiability, and logical consequence

We can now start to discuss the central notions of validity and satisfiability.

Definition 2.2.3. Let φ be a propositional formula.

1. φ is *valid*, or a *tautology*, if for every truth assignment e, $e(\varphi) = \mathbf{T}$.
2. φ is *satisfiable*, if there is a truth assignment e such that $e(\varphi) = \mathbf{T}$.
3. φ is *unsatisfiable*. if for every truth assignment e, $e(\varphi) = \mathbf{F}$.

Definition 2.2.4. Let $\varphi_1, \ldots, \varphi_n, \varphi$ be formulas. We say that φ is a *(logical) consequence* of $\varphi_1, \ldots, \varphi_n$, or that φ *follows* from $\varphi_1, \ldots, \varphi_n$ if for all assignments e such that $e(\varphi_1) = \cdots = e(\varphi_n) = \mathbf{T}$ we also have $e(\varphi) = \mathbf{T}$. In that case we write

$$\{\varphi_1, \ldots, \varphi_n, \varphi\} \models \varphi.$$

We call $\varphi_1, \ldots, \varphi_n$ assumptions, or premises, and φ the conclusion.

Assuming that the empty set $\emptyset$ is valid we sometimes write $\models \varphi$ to say that a formula φ is valid.

Recall our example from the start of the chapter: "Whenever it rains, the grass is wet. The grass is dry. Therefore, it is not raining." We can now see why we said that the argument is correct since we can conclude that it is not raining from our assumptions.

Proposition 2.2.3. *The formula $\neg A$ is a consequence of $\neg B$ and $A \to B$.*

Proof. We write down the truth table for $A \to B$, $\neg B$ and $\neg A$:

A	B	$A \to B$	$\neg B$	$\neg A$
T	**T**	**T**	**F**	**F**
T	**F**	**F**	**T**	**F**
F	**T**	**T**	**F**	**T**
F	**F**	**T**	**T**	**T**

We can see that the only assignment making all assumptions true is $e(A) = \mathbf{F}$, $e(B) = \mathbf{F}$. Under this assignment the conclusion $e(\neg A)$ is also true. Therefore, $\{A \to B, \neg B\} \models \neg A$. □

Definition 2.2.5. A formula φ is *logically equivalent* (sometimes also called truth equivalent) to a formula ψ if $\{\varphi\} \models \psi$ and $\{\psi\} \models \varphi$. We write

$$\varphi \equiv \psi.$$

Exercise 2.2.2. Let φ be a propositional formula. Then

$$\varphi \equiv \neg\neg\varphi. \qquad \text{(law of double negation)}$$

Proposition 2.2.4. *A propositional formula φ is valid if and only if $\neg\varphi$ is unsatisfiable.*

Proof. First assume that φ is valid and that e is a truth assignment. Then $e(\varphi) = \mathbf{T}$. Therefore, by Definition 2.2.2, $e(\neg\varphi) = \mathbf{F}$.

Now, assume that $\neg\varphi$ is unsatisfiable and that e is a truth assignment. Then $e(\neg\varphi) = \mathbf{F}$, and therefore $e(\neg\neg\varphi) = \mathbf{T}$. But $\neg\neg\varphi \equiv \varphi$ and thus $e(\varphi) = e(\neg\neg\varphi) = \mathbf{T}$. □

Let φ and θ be propositional formulas and suppose $\psi \in sub(\varphi)$. We let $\varphi[\psi/\theta]$ be the formula obtained by replacing every occurrence of ψ in φ by θ.

Exercise 2.2.3. Let φ and θ be propositional formulas and $\psi \in sub(\varphi)$. Show that $\varphi[\psi/\theta]$ is a propositional formula.

Theorem 2.2.5. *Let φ be a propositional formula and $\psi \in sub(\varphi)$. Suppose $\hat{\psi} \equiv \psi$. Then $\varphi[\psi/\hat{\psi}] \equiv \varphi$.*

Proof. We prove this by induction on propositional formulas. First, if $\varphi = P$ for a propositional variable P, then $sub(\varphi) = \{P\}$ and thus $\psi = \varphi = P = \hat{\psi}$. Therefore, trivially $\varphi \equiv \varphi[\psi/\hat{\psi}]$.

Now assume that $\varphi = \neg\theta$ and the theorem holds for θ in place of φ. Then either $\psi = \varphi$ or $\psi \in sub(\theta)$. In the first case, $\varphi = \psi \equiv \hat{\psi} = \varphi[\psi/\hat{\psi}]$ and

thus the result holds. In the second case, we have by induction hypothesis that $\theta \equiv \theta[\psi/\hat{\psi}]$. Let e be any truth assignment, then

$$\begin{aligned} e(\varphi) = e(\neg\theta) = \mathbf{T} &\text{ iff } e(\theta) = \mathbf{F} && \text{(by definition of } \neg) \\ &\text{iff } e(\theta[\psi/\hat{\psi}]) = \mathbf{F} && \text{(by hypothesis)} \\ &\text{iff } e(\neg\theta[\psi/\hat{\psi}]) = \mathbf{T}. && \text{(by definition of } \neg) \end{aligned} \tag{2.1}$$

Thus, we have that $\varphi = \neg\theta \equiv \neg\theta[\psi/\hat{\psi}] = \varphi[\psi/\hat{\psi}]$ as required.

At last, assume that $\varphi = (\theta \to \chi)$ and the theorem holds both for θ and χ in place of φ. Then either $\psi = \varphi$ in which case again $\varphi = \psi \equiv \hat{\psi} = \varphi[\psi/\hat{\psi}]$, or $\psi \in sub(\theta) \cup sub(\chi)$. In the latter case we have that $\varphi[\psi/\hat{\psi}] = (\theta[\psi/\hat{\psi}] \to \chi[\psi/\hat{\psi}])$. Now for any truth assignment e,

$$\begin{aligned} e(\varphi) = \mathbf{F} &\text{ iff } e(\theta) = \mathbf{T} \text{ and } e(\chi) = \mathbf{F} && \text{(by definition of } \to) \\ &\text{iff } e(\theta[\psi/\hat{\psi}]) = \mathbf{T} \text{ and } e(\chi[\psi/\hat{\psi}]) = \mathbf{F} && \text{(by hypothesis)} \\ &\text{iff } e((\theta[\psi/\hat{\psi}] \to \xi[\psi/\hat{\psi}])) = \mathbf{F} && \text{(by definition of } \to) \end{aligned}$$

Thus, we get $\varphi \equiv \varphi[\psi/\hat{\psi}]$ as required. □

2.2.3 Other connectives

So far our logic has connectives with the meaning "implies that" and "it is not the case that". At first sight this seems to weak to formally capture even simple sentences such as

> "Mary likes cake and Hugo bakes cake."

We might therefore be tempted to extend the syntax and semantics of propositional logic to allow additional symbols capturing the meaning of "and", "or", or "if and only if". However this is not necessary. We will see that the connectives capturing these are logically equivalent to formulas using only $\neg$ and $\to$. Towards this let $\wedge$ have the meaning "and", $\vee$ have the meaning "or" and $\leftrightarrow$ have the meaning "if and only if", i.e., were we to define the semantics $\wedge$, $\vee$, and $\leftrightarrow$ we would require

$$e((\varphi \wedge \psi)) = \begin{cases} \mathbf{T} & \text{if } e(\varphi) = \mathbf{T} \& e(\psi) = \mathbf{T} \\ \mathbf{F} & \text{otherwise} \end{cases},$$

$$e((\varphi \vee \psi)) = \begin{cases} \mathbf{T} & \text{if } e(\varphi) = \mathbf{T} \text{ or } e(\psi) = \mathbf{T} \\ \mathbf{F} & \text{otherwise} \end{cases},$$

$$e((\varphi \leftrightarrow \psi)) = \begin{cases} \mathbf{T} & \text{if } e((\varphi \rightarrow \psi)) = \mathbf{T} \text{ and } e((\psi \rightarrow \varphi)) = \mathbf{T} \\ \mathbf{F} & \text{otherwise} \end{cases}.$$

Theorem 2.2.6. *Let φ and ψ be propositional formulas. Then*

1. $(\varphi \wedge \psi) \equiv \neg(\varphi \rightarrow \neg\psi)$,
2. $(\varphi \vee \psi) \equiv (\neg\varphi \rightarrow \psi)$,
3. $(\varphi \leftrightarrow \psi) \equiv \neg((\varphi \rightarrow \psi) \rightarrow \neg(\psi \rightarrow \varphi))$.

Proof. We prove (1) by writing down a truth table and leave (2) and (3) as an exercise.

φ	ψ	$(\varphi \wedge \psi)$	$\neg\psi$	$\neg(\varphi \rightarrow \neg\psi)$
T	**T**	**T**	**F**	**T**
T	**F**	**F**	**T**	**F**
F	**T**	**F**	**F**	**F**
F	**F**	**F**	**T**	**F**

□

Definition 2.2.6. A formula of the form $(\varphi \wedge \psi)$ is called a *conjunction of φ and ψ*. A formula of the form $(\varphi \vee \psi)$ is called a *disjunction of φ and ψ*. The formulas φ and ψ are called *conjuncts* or *disjuncts*, respectively.

Exercise 2.2.4. Complete the proof of Theorem 2.2.6.

From now on we will use $\wedge$, $\vee$, and $\leftrightarrow$ in our formulas and treat them as if they were part of our language. This is justified by Theorem 2.2.6.

Exercise 2.2.5. Formally define symbols $\times$ and $\star$ capturing the meaning of "A or B but not both" and "A and B can not both be true" respectively. Find formulas φ and ψ in propositional logic such that $A \times B \equiv \varphi$ and $A \star B \equiv \psi$. (You may use $\wedge$, $\vee$, $\leftrightarrow$ in your proofs)

Proposition 2.2.7. *Let φ, ψ, and θ be propositional formulas. The following equivalences hold:*

$$((\varphi \wedge \psi) \wedge \theta) \equiv (\varphi \wedge (\psi \wedge \theta)) \qquad \text{(Associativity-}\wedge\text{)}$$
$$((\varphi \vee \psi) \vee \theta) \equiv (\varphi \vee (\psi \vee \theta)) \qquad \text{(Associativity-}\vee\text{)}$$
$$(\varphi \wedge \psi) \equiv (\psi \wedge \varphi) \qquad (\varphi \vee \psi) \equiv (\psi \vee \varphi)$$

Proof. We consider the truth tables and check that the respective columns are the same.

φ	ψ	θ	$(\varphi \wedge \psi)$	$((\varphi \wedge \psi) \wedge \theta)$	$(\psi \wedge \theta)$	$(\varphi \wedge (\psi \wedge \theta))$
T	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**F**	**F**	**F**	**F**
T	**T**	**F**	**T**	**F**	**F**	**F**
T	**F**	**F**	**F**	**F**	**F**	**F**
F	**T**	**T**	**F**	**F**	**F**	**F**
F	**F**	**T**	**F**	**F**	**F**	**F**
F	**T**	**F**	**F**	**F**	**F**	**F**
F	**F**	**F**	**F**	**F**	**F**	**F**

φ	ψ	θ	$(\varphi \vee \psi)$	$((\varphi \vee \psi) \vee \theta)$	$(\psi \vee \theta)$	$(\varphi \vee (\psi \vee \theta))$
T	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**T**
F	**T**	**T**	**T**	**T**	**T**	**T**
F	**F**	**T**	**F**	**T**	**T**	**T**
F	**T**	**F**	**T**	**T**	**T**	**T**
F	**F**	**F**	**F**	**F**	**F**	**F**

The last two equivalences are easy to see from the truth tables. □

From now on we will make the convention that we may omit parenthesis in disjunctions and conjunctions, e.g., instead of $(\varphi \wedge (\varphi \wedge \theta))$ we may write $(\varphi \wedge \varphi \wedge \theta)$. From a point of view of semantics this is warranted by Proposition 2.2.7. From a syntactical point of view of course $(\varphi \wedge \varphi \wedge \theta)$ is not a formula and there are more than one option to translate it into a logical equivalent formula, depending on where we set the parenthesis. However, we accept this slight abuse of syntax in favor of better readability of formulas. Be aware however, that we can not do this for $\rightarrow$ and if we have different connectives in a formula!

Notice, that even when we have a conjunction $(\ldots((\varphi_1 \wedge \varphi_2) \wedge \cdots \wedge \varphi_n)$ involving more than three conjuncts by Proposition 2.2.7 we may drop the parenthesis. To see this iteratively replace all subformulas of the form $((\varphi \wedge \psi) \wedge \theta)$ by $(\varphi \wedge \psi \wedge \theta)$. After $n-2$ iterations we obtain $(\varphi_1 \wedge \cdots \wedge \varphi_n)$. Of course we use the conventions outlined in the above two paragraphs also for $\vee$.

Exercise 2.2.6. Show that the following holds for formulas φ, ψ and θ:

1. $((\varphi \wedge \psi) \vee \theta) \not\equiv (\varphi \wedge (\psi \vee \theta))$
2. $((\varphi \rightarrow \psi) \rightarrow \theta) \not\equiv (\varphi \rightarrow (\psi \rightarrow \theta))$

2.2.4 Adequate sets of connectives

We have seen that the connectives $\rightarrow$ and $\neg$ are sufficient to obtain, up to logical equivalence, all formulas using connectives $\neg$, $\rightarrow$, $\wedge$, $\vee$, and $\leftrightarrow$. Can we use other connectives to obtain all the formulas?

Definition 2.2.7. A set C of connectives is *adequate* if every formula is logically equivalent to a formula using only connectives from C.

So far we have seen that $\{\neg, \rightarrow\}$ is an adequate set of connectives.

Example 2.2.8. $\{\neg, \vee\}$ *is an adequate set of connectives.*

Proof. It suffices to find for any formula φ, ψ, a formula using only $\neg$ and $\vee$ which is logically equivalent to $(\varphi \to \psi)$. Given φ and ψ, consider the formula $(\neg\varphi \vee \psi)$.

φ	ψ	$(\varphi \to \psi)$	$(\neg\varphi \vee \psi)$
T	**T**	**T**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**T**

It is easy to see from the table that $(\varphi \to \psi) \equiv (\neg\varphi \vee \psi)$ and thus we have found the required formula. □

Example 2.2.9. $\{\neg\}$ *is not an adequate set of connectives.*

Proof. Let φ using $\neg$ as its only connective. Then $\varphi = \neg \ldots \neg P$ for some proposition P. Let n be the number of $\neg$ in φ. Then for any truth assignment e,

$$e(\varphi) = \begin{cases} e(P) & \text{if } n \text{ even,} \\ e(\neg P) & \text{otherwise.} \end{cases}$$

Now let $\psi = P \to Q$ and assume towards a contradiction that $\psi \equiv \varphi$. Let e be a truth assignment such that $e(P) = e(Q) = \mathbf{T}$. Then $e(\psi) = \mathbf{T}$ and since $\psi \equiv \varphi$ also $e(\varphi) = \mathbf{T}$. Let $\hat{e}$ be the truth assignment inverting e, i.e., for all propositions R

$$\hat{e}(R) = e(\neg R).$$

Then $\hat{e}(\psi) = \mathbf{T}$ by the definition of $\to$. Recall that φ is of the form $\neg \ldots \neg P$. As $e(\varphi) = e(P) = \mathbf{T}$ there must be an even number of $\neg$ in φ. By definition $\hat{e}(\varphi) = e(\varphi[^P/_{\neg P}])$ and clearly $\varphi[^P/_{\neg P}]$ has an odd number of negation symbols. So, by our observation above, $\hat{e}(\varphi) = e(\varphi[^P/_{\neg P}]) = e(\neg P) = \mathbf{F}$. This contradicts our assumption that $\varphi \equiv \psi$. □

2.3 SAT and Resolution

2.3.1 Conjunctive and disjunctive normal form

The goal of this section is to develop an algorithm that tests whether a formula is satisfiable or not. We have already informally described and used one such algorithm: truth tables. Given a formula φ we can calculate $sub(\varphi)$ and then calculate its truth table. We then only have to search through the rows to see if there is one truth assignment e such that $e(\varphi) = \mathbf{T}$. However, truth tables quickly become very large and therefore this algorithm is not useful in practice.

We will therefore study another algorithm, called resolution and due to Putnam and Robinson. While this algorithm works better in practice it has the drawback that it only accepts formulas in a certain "normal form", conjunctive normal form, as input.

Definition 2.3.1. A *literal* is a formula of the form P or $\neg P$ where P is a propositional variable.

Definition 2.3.2. A formula φ is in *conjunctive normal form*, short *CNF*, if $\varphi = (C_1 \wedge \cdots \wedge C_n)$ where for each $1 \leq k \leq n$ $C_k = (l_k^1 \vee \cdots \vee l_k^{m(k)})$ where for each $1 \leq i \leq m(k)$, l_k^i is a literal. We refer to the C_k as *clauses* or *CNF constituents* of φ.

In other words, a formula is in conjunctive normal form, if it is the conjunction of disjunctions of literals. Notice that in the above definition both $m(k)$ or n could be 1. Therefore, also the formulas $(l_1 \wedge l_2)$, $(l_1 \vee l_2)$, and l_1 where l_1 and l_2 are literals are in conjunctive normal form.

We will prove that every formula has a conjunctive normal form. We will do this by giving an algorithm which at each step replaces a given formula φ by a logically equivalent formula until it obtains a formula logically equivalent to φ in conjunctive normal form. We will make use of

the following logical equivalences for propositional formulas φ, ψ, and θ:

$$\begin{aligned}
(\varphi \to \psi) &\equiv (\neg\varphi \vee \psi) && \text{(Elimination } \to\text{)}\\
(\varphi \leftrightarrow \psi) &\equiv ((\varphi \to \psi) \wedge (\psi \to \varphi)) && \text{(Elimination } \leftrightarrow\text{)}\\
\neg(\varphi \wedge \psi) &\equiv (\neg\varphi \vee \neg\psi) && \text{(De Morgan's law } \wedge\text{)}\\
\neg(\varphi \vee \psi) &\equiv (\neg\varphi \wedge \neg\psi) && \text{(De Morgan's law } \vee\text{)}\\
(\varphi \wedge (\psi \vee \theta)) &\equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \theta)) && \text{(Distributivity } \wedge\text{)}\\
(\varphi \vee (\psi \wedge \theta)) &\equiv ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) && \text{(Distributivity } \vee\text{)}\\
\varphi &\equiv \neg\neg\varphi && \text{(Law of double negation)}
\end{aligned}$$

Exercise 2.3.1. Prove that the above equivalences hold.

Theorem 2.3.1. *For every propositional formula φ with propositional variables $P_1, \ldots, P_n$, there is a formula ψ in conjunctive normal form with the same propositional variables such that $\varphi \equiv \psi$.*

Proof sketch. We prove this by giving an algorithm that takes any formula to a formula in conjunctive normal form. The algorithm can be seen in Algorithm 1.

We have to verify that for any propositional formula φ, CNF(φ) terminates (i.e., does not get stuck in an endless loop), produces a formula in conjunctive normal form and that $\varphi \equiv \text{CNF}(\varphi)$. To see that the algorithm terminates just notice that φ contains at most finitely many subformulas of the types in the conditions of the above while statements. Therefore each of the while statements must terminate and thus the whole procedure must terminate. Furthermore, by Theorem 2.2.5 we have that for every ψ and $\theta \equiv \psi$, $\varphi \equiv \varphi[\psi/\theta]$. Thus, also $\varphi \equiv CNF(\varphi)$.

It remains to show that $CNF(\varphi)$ is in conjunctive normal form. First, notice that $CNF(\varphi)$ does not contain the connectives $\leftrightarrow$ and $\to$. Furthermore negations only occur in front of propositional variables. Now, assume that $CNF(\varphi)$ is not in conjunctive normal form. Then by the above it contains a subformula of the form $\psi \vee (\theta \wedge \chi)$ but this can not be

Algorithm 1 CNF(φ)

while $(\psi \leftrightarrow \theta) \in sub(\varphi)$ **do** ▷ eliminate all occurrences of $\leftrightarrow$
  $\varphi := \varphi[{}^{(\psi \leftrightarrow \theta)}/_{((\psi \rightarrow \theta) \wedge (\theta \rightarrow \psi))}]$
end while
while $(\psi \rightarrow \theta) \in sub(\varphi)$ **do** ▷ eliminate all occurrences of $\rightarrow$
  $\varphi := \varphi[{}^{(\psi \rightarrow \theta)}/_{(\neg\psi \vee \theta)}]$
end while
while $\xi := \neg(\psi \wedge \theta) \in sub(\varphi)$ or $\xi := \neg(\psi \vee \theta) \in sub(\varphi)$ **do** ▷ push negation in
  if $\xi = \neg(\psi \wedge \theta)$ **then**
    $\varphi := \varphi[{}^{\neg(\psi \wedge \theta)}/_{(\neg\psi \vee \neg\theta)}]$
  else
    $\varphi := \varphi[{}^{\neg(\psi \vee \theta)}/_{(\neg\psi \wedge \neg\theta)}]$
  end if
end while
while $\xi := (\psi \vee (\theta \wedge \chi)) \in sub(\varphi)$ or $\xi := ((\psi \wedge \theta) \vee \chi) \in sub(\varphi)$ **do** ▷ push $\vee$ in using distr. $\vee$
  if $\xi = (\psi \vee (\theta \wedge \chi))$ **then**
    $\varphi := \varphi[{}^{(\psi \vee (\theta \wedge \chi))}/_{((\psi \vee \theta) \wedge (\psi \vee \chi))}]$
  else
    $\varphi := \varphi[{}^{((\psi \wedge \theta) \vee \chi)}/_{((\psi \vee \chi) \wedge (\chi \vee \theta))}]$
  end if
end while
while $\neg\neg\psi \in sub(\varphi)$ **do** ▷ ensure that no $\neg\neg$ is occurring
  $\varphi := \varphi[{}^{\neg\neg\psi}/_{\psi}]$
end while
return φ

the case as it would have been substituted in one iteration of the second to last while loop. To see that the propositional variables are the same just notice that the propositional variables are preserved by the substitutions in the algorithm. □

Example 2.3.2. *Find the CNF of* $\neg(P \to (Q \wedge R))$.

Solution. We proceed according to Algorithm 1.

$$\begin{aligned} \neg(P \to (Q \wedge R)) &\equiv \neg(\neg P \vee (Q \wedge R)) && \text{(Def. } \to\text{)} \\ &\equiv (P \wedge \neg(Q \wedge R)) && \text{(De Morgan's law and } \neg\neg\text{)} \\ &\equiv (P \wedge (\neg Q \vee \neg R)) && \text{(De Morgan's law)} \end{aligned}$$

◀

Example 2.3.3. *Find the CNF of* $((P \wedge Q) \vee (R \wedge S))$.

Solution.

$$\begin{aligned} ((P \wedge Q) \vee (R \wedge S)) &\equiv ((P \wedge Q) \vee R) \wedge ((P \wedge Q) \vee S) && \text{(Distributivity } \vee\text{)} \\ &\equiv ((P \vee R) \wedge (Q \vee R)) \wedge ((P \vee S) \wedge (Q \vee S)) \\ & && \text{(Distributivity } \vee\text{)} \end{aligned}$$

◀

Similar to conjunctive normal form we can define the dual notions *disjunctive normal form*.

Definition 2.3.3. A formula φ is in *disjunctive normal form*, short *DNF*, if $\varphi = (C_1 \vee \cdots \vee C_n)$ where for each $1 \leq k \leq n$ $C_k = (l_k^1 \wedge \cdots \wedge l_k^{m(k)})$ where for each $1 \leq i \leq m(k)$, l_k^i is a literal. We refer to the C_k as the *conjunctions* of φ.

We can prove a similar theorem to Theorem 2.3.1 for disjunctive normal forms of formulas. The proof is symmetric to the one for Theorem 2.3.1. We thus leave it as an exercise.

Exercise 2.3.2. For every propositional formula φ with propositional variables $P_1, \ldots, P_n$, there is a formula ψ in disjunctive normal form with the same propositional variables such that $\varphi \equiv \psi$.

The advantage of formulas in disjunctive normal form is that it is much simpler to check whether a truth assignment satisfies them. If we are given a truth assignment e and a formula φ in disjunctive normal form it is easy to verify whether $e(\varphi) = \mathbf{T}$. For instance if $\varphi = C_1 \vee \cdots \vee C_n$, then by the semantics of disjunction we know that $e(\varphi) = \mathbf{T}$ if and only if there is an $i \leq n$ such that $e(C_i) = \mathbf{T}$. But by the semantics of conjunction $e(C_i) = \mathbf{T}$ if and only if for all $k \leq m(i)$, $e(l_i^k) = \mathbf{T}$. So we just have through the C_i and check whether e makes all literals in C_i true.

Example 2.3.4. *Let the truth assignment e be defined by $e(P) = \mathbf{T}$,$e(Q) = \mathbf{F}$, $e(R) = \mathbf{T}$ and*

$$\varphi = (P \wedge Q \wedge \neg P) \vee (P \wedge \neg R) \vee (P \wedge \neg Q).$$

Solution. It is easy to see that $e((P \wedge Q \wedge \neg P)) = \mathbf{F}$ because $e(\neg P) = \mathbf{F}$. Likewise, $e((P \wedge \neg R)) = \mathbf{F}$ because $e(\neg R) = \mathbf{F}$. On the other hand $e((P \wedge \neg Q)) = \mathbf{T}$ because $e(P) = \mathbf{T}$ and $e(\neg Q) = \mathbf{T}$. So $e(\varphi) = \mathbf{T}$. ◀

Notice that while it is conceptually much simpler to test whether a truth assignment satisfies a formula in disjunctive normal form, in some cases we still have to check for every conjunction in the formula until we can verify that e satisfies the formula.

Conjunctive normal form features a property which is dual to that of disjunctive normal form in the sense that it is simple to check whether a truth assignment does not satisfy a formula in conjunctive normal form. Given e and φ just check clause by clause whether e satisfies it. If we find a clause that is not satisfied by e, then we immediately know that $e(\varphi) = \mathbf{F}$.

Example 2.3.5. *Let the truth assignment e be defined by $e(P) = \mathbf{T}$,$e(Q) = \mathbf{F}$, $e(R) = \mathbf{T}$ and*

$$\varphi = (P \vee Q \vee \neg P) \wedge (\neg P \vee \neg R) \wedge (P \vee \neg Q).$$

Solution. We first look at the first clause and see that $e((P \vee Q \vee \neg P)) = \mathbf{T}$ because $e(P) = \mathbf{T}$, so we check the next clause. We see that $e((\neg P \vee \neg R)) = \mathbf{F}$ because $e(\neg P) = e(\neg R) = \mathbf{F}$. ◀

2.3.2 A short excursion to computer science

In the field of computer science called *complexity theory* one studies the "complexity" of computational problems. A computational problem is given by a set of instances together with a solution for every instance. A special case of computational problems are decision problems, problems where the solution to an instance can either be "yes" or "no". More formally, a computational problem is defined as follows.

Definition 2.3.4. A *computational problem* is given by a set of instances I, a set of solutions S and a function $f : I \to S$.

Definition 2.3.5. A *decision problem* is a computational problem where $S = \{yes, no\}$.

Complexity theorists are interested in the complexity of the function f. It will become clear what we mean by that when we discuss the following example.

One, if not the most famous decision problem in complexity theory is **SAT**, the problem of the satisfiability of propositional formulas in conjunctive normal form. More formally, **SAT** is defined as follows:

SAT: The set of instances of **SAT** are propositional formulas in conjunctive normal form and the function f is the function mapping a formula φ in CNF to "yes" if φ is satisfiable and to "no" if it is unsatisfiable.

We have already seen that the problem **SAT** can be solved by an algorithm, even for all formulas, not only those in conjunctive normal form – *truth tables*. First write down all the possible truth assignments for the variables in φ. Then we use the semantics of propositional logic to fill out the rest of the table row by row. As soon as you find a row in which φ is true stop and return *yes*. If no such row exists answer *no*. So we know that there is an algorithm for the function f.

The algorithm we developed has a draw back, namely that it is not very efficient. As the size of the input formula increases the time it takes to check whether the formula is true increases dramatically. This is true even for formulas in conjunctive normal form.

Assume that n is the number of clauses in the formula φ in conjunctive normal form. We write down a column for every variable and clause in φ. Now we start by filling out the rows, first filling out the values for the variables, and then computing the variables for the clauses. We count every cell containing a clause we fill out as a computation step. We may stop after we have finished computing a row where all the clauses are true. But if φ is unsatisfiable then we have to go through all the rows. Assume that the number of variables in φ is equal to the number of clauses n, then there are 2^n distinct truth assignments and we will need $2^n \cdot n$ computation steps to find the solution. So, in this case the runtime (number of computation steps) of our algorithm is exponential in the size of the input. An algorithm which is exponential can not be considered an efficient algorithm since already for small n, 2^n becomes unfeasible large even for the most modern supercomputers. If we have a formula with 300 variables that is unsatisfiable, then we would have to check 2^{300} rows, and this is larger than the number of atoms in the observable universe (estimated to be 10^{82}). Clearly a ridiculous amount of time which even on the fastest supercomputers is unfeasible.

In complexity theory we want to classify computational problems given their runtime into classes called complexity classes. One of the most important class of problems is the class P, the class of problems which have *polynomial time algorithms* – algorithms whose runtime is bound by a polynomial of the input, for example, n^2, or $n^3 + 10$. Polynomial time algorithms are generally considered to be efficient.

Another important class of problems is the class NP. The name NP stands for **n**ondeterministic **p**olynomial time. Problems in NP are characterized by admitting the so called *guess & check* approach to solve them. For problems in NP it is unknown whether a polynomial time algorithm exists, but there exists a polynomial time procedure for instances with positive solutions that makes a nondeterministic *guess*, and then if it made

the correct guess it will output *yes* in polynomial time. Of course the guess is random, so we might as well be wrong a lot of times before making a good guess and thus do not have an algorithm which is efficient in general.

The problem **SAT** is clearly in NP. Given a formula φ in CNF randomly choose a truth assignment (guess part). Once we have chosen the truth assignment we can *check* whether it makes φ true in polynomial time (in fact linear time). So if φ was satisfiable and the algorithm guessed the correct truth assignment, then it is able to answer *yes* in polynomial time.

Also, clearly $\mathsf{P} \subseteq \mathsf{NP}$. One of the most famous open questions in computer science is whether this inclusion is strict, i.e., whether $\mathsf{P} = \mathsf{NP}$. The problem **SAT** is in another important class of problems, the class of NP-hard problems. A problem is NP-*hard* if for every $P \in \mathsf{NP}$ there exists a polynomial time algorithm Φ, also called a reduction, that takes as input an instance i of P and outputs an instance $\Phi(i)$ such that the solution of i is *yes* if and only if the solution for $\Phi(i)$ is *yes*. This implies that if we have found a polynomial time algorithm for **SAT**, then we would get a polynomial time algorithm for every problem in NP. Thus we would have shown that $\mathsf{P} = \mathsf{NP}$.

A problem P is NP-complete if it is NP-hard and $P \in \mathsf{NP}$. Not only is **SAT** NP-complete, it was also the first problem known to be NP-complete and most of the problems which are nowadays known to be NP-complete have been shown to be NP-complete by giving a reduction from **SAT**. If there is a reduction from **SAT** to P, then P is NP-hard since reductions are transitive.

Notice that just because a problem is in NP and thus the best known algorithms for it do not run in polynomial time does not mean that we can not solve this problem efficiently in practice. It just means that for some instances in that problem we will need exponential time, even with the best known algorithms. But for instance, for **SAT** we know algorithms which work very well in practice and will check satisfiability of propositional formulas within very little time in most cases.

2.3.3 Satisfiability revisited

In Section 2.2.2 we defined what it means for a formula φ to be satisfiable. We can extend this definition to sets of formulas.

Definition 2.3.6. Let Γ be a set of formulas. We say that Γ is *satisfiable*, if there is a truth assignment e such that $e(\gamma) = \mathbf{T}$ for all $\gamma \in \Gamma$. We may write $e(\Gamma) = \mathbf{T}$ if $e(\gamma) = \mathbf{T}$ for all $\gamma \in \Gamma$. If Γ is not satsifiable, then we say that Γ is unsatisfiable.

Proposition 2.3.6. *Let Γ be a set of formulas and let φ be a formula. Then the following holds.*

1. *If Γ is satisfiable and $\Gamma \models \varphi$, then $\Gamma \cup \{\varphi\}$ is satisfiable.*
2. *$\Gamma \not\models \varphi$ if and only if $\Gamma \cup \{\neg\varphi\}$ is satisfiable.*

Proof. The first statement follows from the definition of logical consequence. If Γ is satisfiable then there is e such that $e(\Gamma) = \mathbf{T}$. Since $\Gamma \models \varphi$ this implies that $e(\varphi) = \mathbf{T}$. Thus $e(\Gamma \cup \{\varphi\}) = \mathbf{T}$.

For the second statement we first show the direction from left to right. If $\Gamma \not\models \varphi$ then by definition of logical consequence Γ is satisfiable. Thus there is e such that $e(\Gamma) = \mathbf{T}$. Furthermore, $e(\varphi) = \mathbf{F}$. Hence, $e(\neg\varphi) = \mathbf{T}$ and $e(\Gamma \cup \{\neg\varphi\}) = \mathbf{T}$. For the other direction assume that $\Gamma \cup \{\neg\varphi\}$ is satisfiable. Let e be a assignment satisfying $\Gamma \cup \{\neg\varphi\}$, then $e(\neg\varphi) = \mathbf{T}$ and therefore $e(\varphi) = \mathbf{F}$. As $e(\Gamma) = \mathbf{T}$, the assignment e witnesses that $\Gamma \not\models \varphi$. □

Exercise 2.3.3. Let Γ be a finite set of formulas, i.e., $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$. Then $(\gamma_1 \wedge \cdots \wedge \gamma_n)$ is satisfiable if and only if Γ is satisfiable.

This exercise allows us to consider formulas in conjunctive normal form as a set of clauses. On the other hand, given a clause C, we could also write it as a set of literals without any ambiguity, since disjunction is commutative. We will from now on call such a set a clause set.

Definition 2.3.7. A *clause set* (sometimes also just clause) is a (possibly empty) finite set of literals.

Definition 2.3.8. A set Γ of clause sets is *satisfiable* if and only if there is at least one truth assignments e such that for every clause set $C \in \Gamma$, there is some literal $l \in C$ with $e(l) = \mathbf{T}$.

Example 2.3.7. *The set* $\{\{P, \neg Q\}, \{Q, P\}\}$ *is satisfiable.*

Notice that a set of clauses containing the empty clause (empty set) can not be satisfiable by definition.

Example 2.3.8. *The set* $\{\{P, \neg P\}, \{\}\}$ *is not satisfiable.*

Motivated by the observation preceding Definition 2.3.7 we may define for every formula in conjunctive normal form φ, the set

$$C(\varphi) = \{\{l \in C\} : C \text{ a clause in } \varphi\}.$$

It is now not hard to check that $C(\varphi)$ as a set of clauses is satisfiable if and only if φ is satisfiable. On the other hand we also have that for every finite set of clause sets Γ we can get a formula φ_Γ which is satisfiable if and only if Γ is. If Γ does not contain the empty clause then we can just replace the clause sets with the respective disjunctions. And if we have an empty clause we just let $\varphi_\Gamma = A \wedge \neg A$.

Notice how we have just described a reduction from **SAT** to the problem of satisfying sets of clauses and vice versa. It is not hard to see that these reductions are polynomial time reductions. In fact they are linear in the size of the sets.

2.3.4 The resolution rule and the Davis-Putnam algorithm

We want to find an algorithm that tests sets of propositional formulas in conjunctive normal form for satisfiability. The original algorithm we will discuss is due to Martin Davis and Hilary Putnam. Since it is invention in the 1960 it has seen considerable improvements. However, it is still the core of many modern programs that test formulas for satisfiability (usually called SAT-solvers).

The algorithm is based on the following rule.

Definition 2.3.9. Let $\mathcal{C}_1 = \{a_1, \ldots, a_n, X\}$ and $\mathcal{C}_2 = \{b_1, \ldots, b_m, \neg X\}$ be clauses, then the *resolution rule* is:

$$\frac{\{a_1, \ldots, a_n, X\} \qquad \{b_1, \ldots, b_m, \neg X\}}{\{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_m\}}$$

We call $\{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_m\}$ the *resolvent* of $\mathcal{C}_1$ and $\mathcal{C}_2$ and $(\mathcal{C}_1, \mathcal{C}_2)$ a resolution pair for X.

The rule is to be interpreted as "given a clause set Γ containing $\{a_1, \ldots, a_n, C\}$ and $\{b_1, \ldots, b_m, \neg C\}$ with contradicting literals c, $\neg C$, add the union of the two clauses minus $\{C, \neg C\}$ to the clause set".

We will encounter rules like this at various points of this cause. At the top we always have some premises, the dividing line may be read as "entails" or "infer" and the content after the line is what we may infer from the premises using the rule.

At a single step in our arguments we will not use only one application but rather will do *resolution on* X where X is a variable.

Definition 2.3.10. Let Γ be a set of clauses and X be a propositional variable. To perform *resolution on* X for Γ, do the following:

1. Remove all clauses $\mathcal{C} \in \Gamma$ with $\{X, \neg X\} \subseteq \mathcal{C}$.
2. Apply the resolution rule for all remaining resolution pairs for X in Γ.
3. Remove all clauses $\mathcal{C} \in \Gamma$ with either $X \in \mathcal{C}$ or $\neg X \in \mathcal{C}$.

Example 2.3.9. *Perform resolution on P for*

$$\Gamma = \{\{P, Q, R\}, \{P, \neg R\}, \{P, \neg P, Q\}, \{\neg P, R\}, \{\neg P, Q, \neg S\}, \{Q\}, \{\neg Q, R\}\}$$

Solution. We first remove all the clauses containing both P and $\neg P$. The resulting clause set is

$$\{\{P, Q, R\}, \{P, \neg R\}, \{\neg P, R\}, \{\neg P, Q, \neg S\}, \{Q\}, \{\neg Q, R\}\}.$$

Next we resolve all the resolution pairs for P. We get the following instances of the resolution rules:

$$\frac{\{P,Q,R\} \qquad \{\neg P,R\}}{\{Q,R\}} \qquad \frac{\{P,Q,R\} \qquad \{\neg P,Q,\neg S\}}{\{Q,R,\neg S\}}$$

$$\frac{\{P,\neg R\} \qquad \{\neg P,R\}}{\{\neg R,R\}} \qquad \frac{\{P,\neg R\} \qquad \{\neg P,Q,\neg S\}}{\{\neg R,Q,\neg S\}}$$

So, after application of *resolution on* X we have

$$\Gamma = \{\{Q,R\},\{R,\neg R\},\{Q,R,\neg S\},\{\neg R,Q,\neg S\},\{Q\}\{\neg Q,R\}\}.$$

◀

The *Davis Putnam procedure* (or Davis Putnam algorithm) is now defined as in the Fig. 2.1. Note that we use $\neg l$ to denote the converse of l, i.e., if $l = X$, then $\neg l$ is $\neg X$ and if $l = \neg X$, then $l = X$.

Example 2.3.10. *Use the DPP to decide whether*

$$\{\{P,Q,R\},\{P,\neg R\},\{P,\neg P,Q\},\{\neg P,R\},\{\neg P,Q,\neg S\},\{Q\},\{\neg Q,R\}\}$$

is satisfiable.

Solution. Since $\{P,\neg P,Q\}$ contains both P and $\neg P$ we remove it to obtain:

$$\{\{P,Q,R\},\{P,\neg R\},\{\neg P,R\},\{\neg P,Q,\neg S\},\{Q\},\{\neg Q,R\}\}$$

There is no clause containing S, so we may remove $\{\neg P,Q,\neg S\}$ and obtain:

$$\{\{P,Q,R\},\{P,\neg R\},\{\neg P,R\},\{Q\},\{\neg Q,R\}\}$$

We do resolution on P:

$$\frac{\{\text{P, Q, R}\} \qquad \{\neg\text{P, R}\}}{\{\text{Q,R}\}} \qquad \frac{\{\text{P,}\neg\text{R}\} \qquad \{\neg\text{P, R}\}}{\{\text{R,}\neg\text{R}\}}$$

We obtain

$$\{\{Q,R\},\{R,\neg R\},\{Q\},\{\neg Q,R\}\}$$

We can remove $\{R,\neg R\}$ and get:

$$\{\{Q,R\},\{Q\},\{\neg Q,R\}\}$$

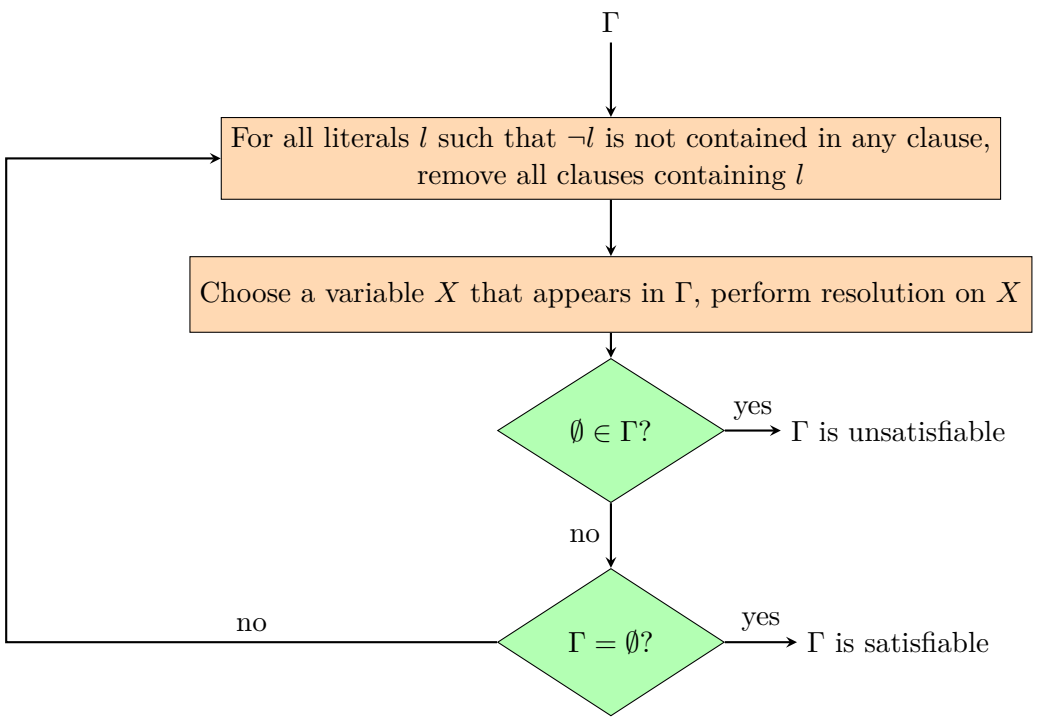

Figure 2.1: The Davis Putnam procedure for propositional logic

As $\neg R$ does not occur in any clauses, we may remove all clause including R and get

$$\{\{Q\}\}$$

We may now remove $\{Q\}$ as $\neg Q$ does not occur and get the empty set $\emptyset$. Therefore the original set is satisfiable.

◀

We still need to show that the Davis Putnam procedure is *correct*, i.e., that Γ is satisfiable if and only if the procedure returns *yes*, and that Γ is unsatisfiable if and only if the procedure returns *no*. Notice that both of these statements are necessary, as it theoretically might be the case that the algorithm gets stuck in a loop and never returns anything as it does not terminate. However, it is not hard to see that the procedure always terminates. Just notice that after each resolution step, the number of variables in Γ reduces by 1. So, as any finite set of clauses only consists of finitely many variables, say n, the procedure is guaranteed to terminate after n many iterations of the loop. However, notice that while resolution on X will remove the clauses containing X, it will introduce new clauses. For instance, assume there are 6 clauses containing X and 6 clauses containing $\neg X$ but that the clauses are disjoint apart from that. Then resolution on X will introduce $6 \cdot 6$ new clauses.

To improve readability let the rule "For all literals l such that $\neg l$ is not contained in any clause, remove all clauses containing l" be from now on called the *unnegated literal removal rule*. (In the lecture we called it rule *)

To show that the Davis Putnam procedure it is sufficient to show that both resolution on X and unnegated literal removal preserve satisfiability.

Lemma 2.3.11. *Unnegated literal removal preserves satisfiability, i.e., if $\hat{\Gamma}$ is the result of applying unnegated literal removal to Γ, then Γ is satisfiable if and only if $\hat{\Gamma}$ is satisfiable.*

Proof. First we show the direction from left to right, i.e., if Γ is satisfiable then so is $\hat{\Gamma}$. Towards that assume that Γ is satisfiable and let e be a truth assignment satisfying Γ. Then for all clauses $\mathcal{C} \in \Gamma$, $e(\mathcal{C}) = \mathbf{T}$. But as unnegated literal removal does not introduce literals we have $\hat{\Gamma} \subseteq \Gamma$ and thus $e(\mathcal{C}) = \mathbf{T}$ for all $\mathcal{C} \in \hat{\Gamma}$. Therefore $\hat{\Gamma}$ is satisfiable.

To see that $\hat{\Gamma}$ is satisfiable implies Γ is satisfiable, assume that $\hat{\Gamma}$ is satisfied by e. Any clause $\mathcal{C} \in \Gamma \setminus \hat{\Gamma}$ has a literal l in it whose negation does not occur in Γ. Furthermore, neither l nor $\neg l$ occur in $\hat{\Gamma}$ and hence we may assume that e is undefined on l. This means that if $l = X$ or

$l = \neg X$, for a propositional variable X, $e(X)$ is not defined. If $l = \neg X$, then set $e(X) = \mathbf{F}$, and if $l = X$, then set $e(X) = \mathbf{T}$. Then $e(\mathcal{C}) = \mathbf{T}$ is true for every $\mathcal{C} \in \Gamma$ containing l. Extending e like this for every l so that $\neg l$ does not occur in Γ clearly gives a satisfying truth assignment for Γ. □

Lemma 2.3.12. *Let Γ be a set of clauses and $\hat{\Gamma}$ be obtained from Γ by resolution on X for some variable X occurring in Γ. Then Γ is satisfiable if and only if $\hat{\Gamma}$ is satisfiable.*

Proof. We first show the direction from left to right, that is if Γ is satisfiable, then so is $\hat{\Gamma}$. Assume that Γ is satisfiable and that e is a truth assignment satisfying Γ. Clauses in $\hat{\Gamma}$ are either of the form $\mathcal{B}$ with $\mathcal{B} \in \Gamma$ where neither $X \in \mathcal{B}$ nor $\neg X \in \mathcal{B}$. Or they are of the form $\mathcal{C} \cup \mathcal{D}$ where $\{X\} \cup \mathcal{C} \in \Gamma$ and $\{\neg X\} \cup \mathcal{D} \in \Gamma$.

If $\mathcal{B}$ is the first type of clause, then we have that there is an $l \in \mathcal{B}$ with $e(l) = \mathbf{T}$ since e satisfies Γ.

If $\mathcal{B}$ is the second type of clause, then we have two cases, either $e(X) = \mathbf{T}$, then there exists $l_1 \in \mathcal{D}$ such that $e(l_1) = \mathbf{T}$ as $e(\mathcal{D} \cup \{\neg X\}) = \mathbf{T}$ because e satisfies Γ. Or $e(X) = \mathbf{F}$, then there exists $l_2 \in \mathcal{C}$ such that $e(l_2) = \mathbf{T}$ as $e(\mathcal{C} \cup \{X\}) = \mathbf{T}$ because e satisfies Γ. In either case $e(\mathcal{C} \cup \mathcal{D}) = \mathbf{T}$ because both $l_1, l_2 \in \mathcal{C} \cup \mathcal{D}$. This shows that $\hat{\Gamma}$ is satisfiable.

Now assume that $\hat{\Gamma}$ is Γ after resolution on X for some variable X and that $\hat{\Gamma}$ is satisfied. Say $\hat{\Gamma}$ is satisfied by e and that $e(X)$ is not defined (We may assume this as X does not occur in $\hat{\Gamma}$). We extend e to satisfy Γ by defining $e(X)$. If $\mathcal{B} \in \hat{\Gamma}$ and $\mathcal{B} \in \Gamma$, then we have nothing to do since $e(l) = \mathbf{T}$ for a literal in $\mathcal{B}$. Any clause containing X and $\neg X$ in Γ is satisfied, no matter if we set $e(X) = \mathbf{T}$ or $e(X) = \mathbf{F}$. If $\mathcal{B} \in \hat{\Gamma}$ and $\mathcal{B} \notin \Gamma$ then $\mathcal{B} = \mathcal{C} \cup \mathcal{D}$ where $\mathcal{C} \cup \{X\} \in \Gamma$ and $\mathcal{D} \cup \{\neg X\} \in \Gamma$. Let $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and $\mathcal{D}_1, \ldots, \mathcal{D}_m$ be all clauses such that $\mathcal{C}_i \cup \{X\} \in \Gamma$ and $\mathcal{D}_j \cup \{\neg X\} \in \Gamma$ for $i \leq n$, $j \leq m$. We know that for all $\mathcal{C}_i \cup \mathcal{D}_j$, $i \leq n$, $j \leq m$, e satisfies some literal in $\mathcal{C}_i \cup \mathcal{D}_j$. If for every $i \leq n$, e satisfies some literal in $\mathcal{C}_i$, we may set $e(X) = \mathbf{F}$ and get that e satisfies all clauses $\mathcal{D}_i \cup \{\neg X\}$ and thus Γ. Similarly, if e satisfies some literal in $\mathcal{D}_i$ for all $i \leq n$, then we can set $e(X) = \mathbf{T}$ and get that e satisfies Γ.

Now, assume that e does not satisfy a literal in some $\mathcal{C}_j$, $j \leq n$, then $\hat{\Gamma}$ contains

$$\mathcal{C}_j \cup \mathcal{D}_1, \mathcal{C}_j \cup \mathcal{D}_2, \ldots, \mathcal{C}_j \cup D_n$$

and therefore $\mathcal{D}_i$ as well as $\mathcal{D}_i \cup \{\neg X\}$ are satisfied by e. So we can set $e(X) = \mathbf{T}$ and e now satisfies $\mathcal{C}_i \cup \{X\}$, for all $i \leq n$, and in particular $\mathcal{C}_j \cup \{X\}$. It might also be the case that after removing clauses containing both X and $\neg X$ from Γ, we can not find any resolution pairs but that X still occurs in Γ. But this might only happen if X occurs only positively in Γ (i.e., $\neg X$ does not occur in Γ), or if X occurs only negatively (i.e., X does not occur in Γ). In the first case we can just set $e(X) = \mathbf{T}$ and get that every clause in Γ is then satisfied, and in the latter case we can set $e(X) = \mathbf{F}$ and get that every clause is satisfied. Thus also Γ is satisfied by e. □

Theorem 2.3.13. *The Davis Putnam procedure terminates on all inputs and is correct, i.e., DPP*$(\Gamma) =$ *yes iff* Γ *is satisfiable.*

Proof. As discussed in the paragraph above Lemma 2.3.11 DPP terminates on any set of clauses, as the number of variables occurring in Γ is strictly less after an iteration of the loop in the DPP than before.

To show that Γ is correct it is sufficient to show that every step in the algorithm preserves satisfiability of formulas. To see this, notice that when DPP terminates on input Γ, then what remains of Γ is either the emptyset $\emptyset$ or the empty clause is an element of Γ. Let $\hat{\Gamma}$ be the remains of Γ after the DPP terminated. Then if all the steps preserve satisfiability, Γ is satisfiable if and only if $\hat{\Gamma}$ is. Either $\hat{\Gamma}$ is unsatisfiable, then $\emptyset \in \hat{\Gamma}$ and the DPP answers no. Or $\hat{\Gamma}$ is empty. Then DPP answers yes. So the DPP would be correct.

We have already seen in Lemmas 2.3.11 and 2.3.12 that all steps preserve satisfiability. So we can conclude that the Davis Putnam procedure terminates on all inputs and is correct.

□

Notice that by the comments just before Section 2.3.4 we can use the

DPP procedure to check the satisfiability of formulas in CNF and even arbitrary formulas. Given a propositional formula φ:

1. Transform φ into CNF.
2. Get a set of clauses Γ from the CNF of φ.
3. Use the Davis Putnam Procedure to check whether Γ is satisfiable.

Checking satisfiability using only the resolution rule

Note that if Γ is unsatisfiable, then the Davis Putnam procedure will run until it contains the empty clause. What if we were not to remove any clauses from Γ at each step and instead just add the new clauses derivable by the resolution rule to Γ? Then we would have no hope to ever get the empty set but if Γ was unsatisfiable then we would obtain the empty clause. Let us make this idea more formal.

Definition 2.3.11. Let Γ be a set of clauses. Then $Res(\Gamma)$ is the set of resolvents of all resolution pairs in Γ. Furthermore, let

$$Res^0 = \Gamma \text{ and } Res^{n+1}(\Gamma) = Res(Res^n(\Gamma)) \cup Res^n(\Gamma).$$

Theorem 2.3.14. *Let Γ be a finite set of clauses. Then there is $n \in \mathbb{N}$ such that $Res^{n+1}(\Gamma) = Res^n(\Gamma)$. Furthermore, Γ is unsatisfiable if and only if $\emptyset \in Res^n(\Gamma)$.*

Proof. Say there are m variables occuring in Γ. Let var_Γ be the set of variables in Γ. Then, every clause having variables from var_Γ is a subset of

$$var_\Gamma \cup \{\neg P : P \in var_\Gamma\}.$$

Notice that there are only 2^{2m} subsets of this set. So $Res^{2^{2m}+1}(\Gamma) = Res^{2^{2m}}(\Gamma)$. This proves the first statement of the theorem.

It remains to show that Γ is unsatisfiable if and only if $\emptyset \in Res^n(\Gamma)$. We first show the direction from left to right. To see that if Γ is unsatisfiable, then $\emptyset \in Res^n(\Gamma)$ notice that if Δ is a set of clauses and $\hat{\Delta}$ is the result

of applying literal removal and resolution on X for some X to Δ. Then $\hat{\Delta} \subseteq Res(\Delta) \cup \Delta$. Now if Γ is unsatisfiable, then after applying literal removal and resolution on some variable X a finite number, say m, times, DPP obtains the empty clause, i.e., if Γ^m is Γ after m applications of this rule then $\emptyset \in \Gamma^m$ and by induction $\Gamma^m \subseteq Res^m(\Gamma)$. Thus $\emptyset \in \Gamma^m$.

To see that $\emptyset \in Res^n(\Gamma)$ implies that Γ is unsatisfiable first notice that if $\emptyset \in Res^n(\Gamma)$, then by definition $Res^n(\Gamma)$ is unsatisfiable. It is therefore sufficient to show that $Res^n(\Gamma)$ is satisfiable if and only if Γ is satisfiable. We proof this by induction on n. If $n = 0$, the statement is clear. Assume as the hypothesis that the theorem holds for $n-1$, i.e., that $Res^{n-1}(\Gamma)$ is satisfiable if and only if Γ is satisfiable. It is sufficient to show that $Res^n(\Gamma)$ is satisfiable if and only if $Res^{n-1}(\Gamma)$ is satisfiable. First assume that $Res^n(\Gamma)$ is satisfiable. Then, as $Res^{n-1}(\Gamma) \subseteq Res^n(\Gamma)$, $Res^{n-1}(\Gamma)$ is also satisfiable by definition. On the other hand assume that $Res^{n-1}(\Gamma)$ is satisfied by the truth assignment e. If $\mathcal{B} \in Res^n(\Gamma)$ and $\mathcal{B} \notin Res^{n-1}(\Gamma)$ then $\mathcal{B}$ is obtained by an application of the resolution rule. Let $\mathcal{C} \cup \{X\}$ and $\mathcal{D} \cup \{\neg X\}$ be a resolution pair in $Res^{n-1}(\Gamma)$. Then there is a truth assignment e satisfying the two clauses. If $e(X) = \mathbf{T}$, then there is a literal $l \in \mathcal{D}$ with $e(l) = \mathbf{T}$ and therefore the resolvent $\mathcal{C} \cup \mathcal{D}$ is satisfied by e. If $e(X) = \mathbf{F}$, then there is a literal $l \in \mathcal{C}$ with $e(l) = \mathbf{T}$ and therefore $\mathcal{C} \cup \mathcal{D}$ is again satisfied by e. Therefore every clause obtained by resolution is satisfied by e and thus e satisfies $Res^n(\Gamma)$. □

Lets look at some examples.

Example 2.3.15. *The set* $\Gamma = \{\{\neg P\}, \{R, S\}, \{\neg R, P\}\}$ *is satisfiable.*

Solution.

$$
\begin{aligned}
Res^0(\Gamma) &= \Gamma \\
Res^1(\Gamma) &= \{\{\neg P\}, \{R, S\}, \{\neg R, P\}, \{\neg R\}, \{S, P\}\} \\
Res^2(\Gamma) &= Res^1(\Gamma) \cup \{\{S\}\} \\
Res^3(\Gamma) &= Res^2(\Gamma)
\end{aligned}
$$

◀

Since iterating *Res* derives every clause that is derivable by the resolution rule and therefore it will also derive $\emptyset$ it is sometimes easier to just calculate the resolvents we need to derive $\emptyset$ instead of calculating all resolvents. In this case it is often common to picture this process in some kind of tree.

Example 2.3.16. *The set* $\Gamma = \{\{\neg P\}, \{P, Q\}, \{P, \neg Q\}\}$ *is unsatisfiable.*

Solution. It is possible to obtain the emptyset from Γ as can be seen from the following derivation.

$$\dfrac{\dfrac{\{\mathrm{P},\mathrm{Q}\} \qquad \{\mathrm{P},\neg\mathrm{Q}\}}{\{\mathrm{P}\}}\ \text{res. } Q \qquad \{\neg\mathrm{P}\}}{\emptyset}\ \text{res. } P$$

An alternative derivation is:

$$\dfrac{\dfrac{\{\mathrm{P},\mathrm{Q}\} \qquad \{\neg\mathrm{P}\}}{\{\mathrm{Q}\}}\ \text{res. } P \qquad \dfrac{\{\mathrm{P},\neg\mathrm{Q}\} \qquad \{\neg\mathrm{P}\}}{\{\neg\mathrm{Q}\}}\ \text{res. } P}{\emptyset}\ \text{res. } Q$$

◀

2.4 Proof systems for propositional logic

This part of the notes is based on the open logic text:
https://openlogicproject.org/

Logics commonly have both a semantics and a *derivation*, or *proof* system. The semantics concerns concepts such as truth, satisfiability, validity, and entailment. The purpose of proof systems is to provide a purely syntactic method of establishing entailment and validity. They are purely syntactic in the sense that a *proof* (or *derivation* in such a system is a finite syntactic object, usually a sequence (or other finite arrangement) of *sentences formula*s (in the case of propositional logic sentences are just formulas). Good proof systems have the property that any given sequence

or arrangement of sentences or formulas can be verified mechanically to be "correct."

The simplest (and historically first) derivation systems were *axiomatic*. A sequence of formulas counts as a proof in such a system if each individual formula in it is either among a fixed set of "axioms" or follows from formulas coming before it in the sequence by one of a fixed number of "inference rules"—and it can be mechanically verified if a formula is an axiom and whether it follows correctly from other formulas by one of the inference rules. Axiomatic proof systems are easy to describe but proofs in them are hard to read and understand, and are also hard to produce.

Other proof systems have been developed with the aim of making it easier to construct proofs or easier to understand proofs once they are complete. Examples are natural deduction, truth trees, also known as tableaux proofs, and the sequent calculus.

So for a given logic, such as first-order logic, the different proof systems will give different explications of what it is for a sentence to be a *theorem* and what it means for a sentence to be derivable from some others. However that is done, we want these relations to match the semantic notions of validity and entailment. Let's write $\vdash \varphi$ for "φ is a theorem" and "$\Gamma \vdash \varphi$" for "φ is *derivable* (or provable(from Γ." However $\vdash$ is defined, we want it to match up with $\models$, that is:

1. $\vdash \varphi$ if and only if $\models \varphi$
2. $\Gamma \vdash \varphi$ if and only if $\Gamma \models \varphi$

The "only if" direction of the above is called *soundness*. A proof system is sound if provability $\vdash$ guarantees logical entailment $\models$. So, colloquially, it is sound if everything that is derivable in it is true from a point of view of semantics. Every decent proof system has to be sound; unsound proof systems are not useful at all. After all, the entire purpose of a proof is to provide a syntactic guarantee of validity or entailment. We'll prove soundness for the proof systems we present.

The converse "if" direction is also important: it is called *completeness*. A complete proof system is strong enough to show that φ is a theorem

whenever φ is valid, and that $\Gamma \vdash \varphi$ whenever $\Gamma \models \varphi$. Completeness is harder to establish, and some logics have no complete proof systems. Both propositional logic and First-order logic (which we will discuss in Chapter 3) do. Kurt Gödel was the first one to prove completeness for a proof system of first-order logic in his 1929 dissertation.

2.4.1 An axiomatic derivation system

Axiomatic derivation systems are perhaps the simplest proof system for logic. A *derivation* is just a sequence of formulas. To count as a derivation, every formula in the sequence must either be an instance of an axiom, or must follow from one or more formulas that precede it in the sequence by a rule of inference. A derivation derives its last formula.

Axiomatic derivation systems have been invented by Frege and are therefore often called "Frege systems". They are also often referred to as "Hilbert systems". The reason for that is probably because of Hilbert's ambitious planned to develop all of mathematics from a small set of axioms using a formal proof system.

Definition 2.4.1. If Γ is a set of propositional formulas then a *derivation* from Γ is a finite sequence φ_1, ..., φ_n of formulas where for each $i \leq n$ one of the following holds:

1. $\varphi_i \in \Gamma$; or
2. φ_i is an axiom; or
3. φ_i follows from some φ_j (and φ_k) with $j < i$ (and $k < i$) by a rule of inference.

The last clause tells us how to derive new statements from Γ. Rules of inference can be read as if then statements saying "If φ_k and φ_j are in the derivation, then it is legal to have φ_i in the derivation."

A proof system is defined by its axioms, rules of inference and what counts as a derivation. Associated to it are the notions of derivability and theorem.

Definition 2.4.2. A formula φ is *derivable* from Γ, written $\Gamma \vdash \varphi$, if there is a derivation from Γ ending in φ.

Definition 2.4.3. A formula φ is a *theorem* if there is a derivation φ from the $\emptyset$. We write $\vdash \varphi$ if φ is a theorem and $\neg\varphi$ if it is not.

What counts as a correct derivation depends on the rules of inference and axioms we allow. Rules of inference can be read as if then statements.

Our proof system will have one rule of inference.

Definition 2.4.4 (Modus ponens)**.** Let φ and ψ be formulas. Then *modus ponens* is the following rule:

$$\frac{(\varphi \to \psi) \qquad \varphi}{\psi}$$

The rule is to be read as follows: "If we have $(\varphi \to \psi)$ in our derivation and φ in our derivation, then it is legal to have ψ in our derivation".

We still need to give our axioms. We would probably get away with using less axioms but then we would have to write longer proofs. We give the axioms as schemes, saying for example "If θ is a formula of the form

$$((\varphi \wedge \psi) \to \varphi)$$

for some formulas φ and ψ, then it is an axiom". So our set of axioms will be infinite but it will only contain formulas of specific forms. And we can check mechanically (using e.g. a computer or by just investigating a formula ourselves) whether a given formula matches one of the schemes, and therefore is an axiom.

Definition 2.4.5 (Axioms)**.** The set of axioms of our proof system is the

set of all formulas of the form:

$$((\varphi \wedge \psi) \to \varphi) \tag{Ax 1}$$
$$((\varphi \wedge \psi) \to \psi) \tag{Ax 2}$$
$$(\varphi \to (\psi \to (\varphi \wedge \psi))) \tag{Ax 3}$$
$$(\varphi \to (\varphi \vee \psi)) \tag{Ax 4}$$
$$(\varphi \to (\psi \vee \varphi)) \tag{Ax 5}$$
$$((\varphi \to \chi) \to ((\psi \to \chi) \to ((\varphi \vee \psi) \to \chi))) \tag{Ax 6}$$
$$(\varphi \to (\psi \to \varphi)) \tag{Ax 7}$$
$$((\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))) \tag{Ax 8}$$
$$((\varphi \to \psi) \to ((\varphi \to \neg\psi) \to \neg\varphi)) \tag{Ax 9}$$
$$(\neg\varphi \to (\varphi \to \psi)) \tag{Ax 10}$$
$$(\varphi \vee \neg\varphi) \tag{Ax 11}$$
$$((\varphi \wedge \neg\varphi) \to \psi) \tag{Ax 12}$$
$$((\varphi \to (\psi \wedge \neg\psi)) \to \neg\varphi) \tag{Ax 13}$$
$$(\neg\neg\varphi \to \varphi) \tag{Ax 14}$$

Example 2.4.1. $\vdash ((\neg P \vee Q) \to (P \to Q))$.

Proof. The formula $((\neg P \vee Q) \to (P \to Q))$ is not the instantiation of any of our axioms, but if we look at (Ax 6) and instantiate it using $\chi = (P \to Q)$, $\varphi = \neg P$ and $\psi = Q$, then we get that the conclusion of the fourth implication is what we want. So we will try to use (Ax 6), some other axioms and modus ponens to obtain $\vdash ((\neg P \vee Q) \to (P \to Q))$. The derivation looks as follows:
$\vdash ((\neg P \vee Q) \to (P \to Q))$

1.	$((\neg P \to (P \to Q)) \to ((Q \to (P \to Q)) \to ((\neg P \vee Q) \to (P \to Q))))$	(Ax 6)
2.	$(\neg P \to (P \to Q))$	(Ax 10)
3.	$((Q \to (P \to Q)) \to ((\neg P \vee Q) \to (P \to Q)))$	1, 2 MP
4.	$(Q \to (P \to Q))$	(Ax 7)
5.	$((\neg P \vee Q) \to (P \to Q))$	3, 4 MP

□

Example 2.4.2. $\{(\varphi \to \psi), (\psi \to \chi)\} \vdash (\varphi \to \chi)$.

Proof. $\{\varphi \to \psi, \psi \to \chi\} \vdash \varphi \to \chi$ □

1.	$(\varphi \to \psi)$	hyp
2.	$(\psi \to \chi)$	hyp
3.	$((\psi \to \chi) \to (\varphi \to (\psi \to \chi)))$	(Ax 7)
4.	$(\varphi \to (\psi \to \chi))$	2, 3 MP
5.	$((\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)))$	(Ax 8)
6.	$((\varphi \to \psi) \to (\varphi \to \chi))$	4, 5 MP
7.	$(\varphi \to \chi)$	1, 6 MP

Proposition 2.4.3. *If* $\Gamma \vdash (\varphi \to \psi)$ *and* $\Gamma \vdash (\psi \to \chi)$*, then* $\Gamma \vdash (\varphi \to \chi)$.

Proof. Suppose $\Gamma \vdash (\varphi \to \psi)$ and $\Gamma \vdash (\psi \to \chi)$. Then there are derivations $\theta_0, \dots \theta_n = (\varphi \to \psi)$ and $\gamma_0, \dots, \gamma_m = \psi \to \chi$ from Γ. Let the formulas $\xi_3, \dots, \xi_7$ be the formulas from the steps 3.-7. in the derivation in Example 2.4.2. Then

$$\theta_0, \dots \theta_n, \gamma_0, \dots, \gamma_m, \xi_3, \dots, \xi_7 = (\varphi \to \chi)$$

is a derivation of $(\varphi \to \chi)$ from Γ. □

Facts about derivability

In Definitions 3.4.4 and 3.4.5 we defined derivability (provability) and theorems. These notions are the corresponding *proof-theoretic notions* to validity and entailment.

In Definition 2.3.6 we defined what it means for a set of formulas to be satisfiable. The corresponding proof-theoretic notion is *consistency*.

Definition 2.4.6. A set Γ of formulas is *consistent* if and only if there exists φ such that $\Gamma \nvdash \varphi$; it is inconsistent otherwise.

Notice that the notions of consistency and derivability depend on the proof system. If we change the axioms to obtain a different proof system

then we get different consistent sets. For instance, consider the proof system obtained by adding $\varphi \wedge \neg\varphi$ to our axioms. Then by (Ax 12) everything is derivable from this system and thus every set is inconsistent.

Let us proof a few facts about the derivability relation.

Proposition 2.4.4 (Reflexivity)**.** *If $\varphi \in \Gamma$, then $\Gamma \vdash \varphi$.*

Proof. φ is a derivation from Γ ending in φ. □

Proposition 2.4.5 (Monotonicity)**.** *If $\Gamma \subseteq \Delta$ and $\Gamma \vdash \varphi$, then $\Delta \vdash \varphi$.*

Proof. Every derivation from Γ is a derivation from Δ. □

Proposition 2.4.6 (Transitivity)**.** *If $\Gamma \vdash \varphi$ and $\{\varphi\} \cup \Delta \vdash \psi$, then $\Gamma \cup \Delta \vdash \psi$.*

Proof. Suppose $\Gamma \vdash \varphi$ and $\{\varphi\} \cup \Delta \vdash \psi$. Then there exists a derivation from Γ of φ and a derivation from $\{\varphi\} \cup \Delta$ of ψ. The concatenation of these two derivations is a derivation from $\Gamma \cup \Delta$. □

Proposition 2.4.7 (Compactness)**.** *1. If $\Gamma \vdash \varphi$, then there is finite $\Gamma_0 \subseteq \Gamma$ so that $\Gamma_0 \vdash \varphi$.*

2. If every finite subset of Γ is consistent, then Γ is consistent.

Proof. ad (1). If $\Gamma \vdash \varphi$, then there exists a derivation $\theta_0, \ldots, \theta_n = \varphi$ from Γ. Let $\Gamma_0 = \{\theta_i : i \leq n \text{ and } \theta_i \in \Gamma\}$. Then $\theta_0, \ldots, \theta_n$ is also a derivation from Γ_0 and Γ_0 is finite as required.

ad (2). We show the contrapositive: If Γ is inconsistent, then there is a finite subset of Γ that is inconsistent. We will need the following claim the proof of which is left as an exercise.

Claim 2.4.7.1. *A set Γ is inconsistent if and only if $\Gamma \vdash (\varphi \wedge \neg\varphi)$ for some formula φ.*

Now if Γ is inconsistent, then $\Gamma \vdash (\varphi \wedge \neg\varphi)$ by the claim. Then by (1) there exists finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash (\varphi \wedge \neg\varphi)$ and this again implies that Γ_0 is inconsistent. □

We leave the proof of the following proposition as an exercise.

Proposition 2.4.8. *A set of formulas Γ is inconsistent if and only if $\Gamma \vdash \varphi$ and $\Gamma \vdash \neg\varphi$ for some formula φ.*

Soundness of our derivation system

The most essential things any proof system should have are: (1) soundness, everything that is derivable in our system should also be true. (2) Completeness, everything that is true should be derivable in our system.

We will first start proving soundness of our system. Recall that what is a derivation depends on the system. So, in our case, on our axioms and rule of inference.

We first prove that all our axioms are valid and that our rule of inference, modus ponens, preserves truth.

Lemma 2.4.9. *Every formula φ of the form of an axiom defined in Definition 2.4.5 is valid.*

Proof. Exercise. □

Lemma 2.4.10. *Let φ and ψ be formulas. Then $\{\varphi, (\varphi \to \psi)\} \models \psi$.*

Proof. Exercise. □

Theorem 2.4.11 (Soundness)**.** *Let Γ be a set of propositional formulas. If $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.*

Proof. We proof this by induction on the length of the derivation $\varphi_1, \ldots, \varphi_n = \varphi$ from Γ recall the definition of a derivation from Definition 2.4.1. The base case is when $n = 1$. Then the derivation is $\varphi_1 = \varphi$ and φ is either an axiom or $\varphi \in \Gamma$. We have to show that for every truth assignment e so that $e(\gamma) = \mathbf{T}$ for $\gamma \in \Gamma$, $e(\varphi) = \mathbf{T}$. If φ is an axiom then it is valid by Lemma 2.4.9 and thus $e(\varphi) = \mathbf{T}$ for every e. On the other hand if $\varphi \in \Gamma$ then the implication holds trivially.

Suppose that for every ψ that has a derivation from Γ of size less than n, $\Gamma \vdash \psi$ implies $\Gamma \models \psi$. Let φ have a derivation of length n from Γ,

i.e., there is a derivation $\varphi_1, \ldots, \varphi_n = \varphi$. There are three cases. If φ is an axiom or $\varphi \in \Gamma$, then $\Gamma \models \varphi$ by a similar argument as in the base case. If φ is not an axiom or $\varphi \notin \Gamma$, then φ must be derived from φ_i and $\varphi_j = \varphi_i \to \varphi$ using modus ponens for some $i, j < n$. Clearly, $\varphi_1, \ldots, \varphi_i$ and $\varphi_1, \ldots \varphi_j$ are derivations of size less than n, so by hypothesis $\Gamma \models \varphi_i$ and $\Gamma \models (\varphi_i \to \varphi)$. It is now easy to see that for every truth assignment e with $e(\gamma) = \mathbf{T}$ for all $\gamma \in \Gamma$, $e(\varphi) = \mathbf{T}$. Because if $e(\gamma) = \mathbf{T}$ for all $\gamma \in \Gamma$, then $e(\varphi_i) = \mathbf{T}$ and $e((\varphi_i \to \varphi)) = \mathbf{T}$. Thus by semantics of $\to$, $e(\psi) = \mathbf{T}$. □

The deduction theorem

Proving things in our axiomatic calculus can in some cases be cumbersome. The deduction theorem is a handy theorem about our calculus that makes many proves easier.

Theorem 2.4.12 (Deduction theorem)**.** *Let Γ be a set of propositional formulas and φ, ψ be formulas. Then $\Gamma \vdash (\varphi \to \psi)$ if and only if $\Gamma \cup \{\varphi\} \vdash \psi$.*

Proof. The direction from left to right is trivial. If $\Gamma \vdash (\varphi \to \psi)$, then by Proposition 2.4.5 also $\Gamma \cup \{\varphi\} \vdash (\varphi \to \psi)$. Let $\varphi_1, \ldots, \varphi_n = (\varphi \to \psi)$ be a derivation from Γ then we can extend the derivation by

n.	$(\varphi \to \psi)$	
n+1.	φ	hyp
n+2.	ψ	n, n+1 MP

to obtain a derivation of ψ from $\Gamma \cup \{\varphi\}$.

We will proof the direction from right to left by induction on the length of the derivation of ψ from $\Gamma \cup \{\varphi\}$. For the base case of the induction assume that ψ is derivable from $\Gamma \cup \{\varphi\}$ using a derivation of length 1. Then either ψ is an axiom or $\psi \in \Gamma \cup \{\varphi\}$. If $\psi \in \Gamma$ or ψ is an axiom then the following is a derivation of $(\varphi \to \psi)$ from Γ:

1.	ψ	hyp or axiom
2.	$(\psi \to (\varphi \to \psi))$	(Ax 7)
3.	$(\varphi \to \psi)$	1, 2 MP

If $\psi = \varphi$, then we have to show that $\Gamma \vdash (\varphi \to \varphi)$ which is an assignment. For the induction step suppose a derivation of ψ from $\Gamma \cup \{\varphi\}$ ends with ψ justified by modus ponens (If not, then either $\psi = \varphi$, or ψ is an axiom and we can reason as in the base case). Then some previous steps in the derivation are $(\chi \to \psi)$ and χ for some formula χ. Hence $\Gamma \cup \{\varphi\} \vdash (\chi \to \psi)$ and $\Gamma \cup \{\varphi\} \vdash \chi$ with shorter derivations (the derivation of ψ cut off at the respective formulas), so the inductive hypothesis applies to them. So we have:

$$\Gamma \vdash (\varphi \to (\chi \to \psi))$$
$$\Gamma \vdash (\varphi \to \chi)$$

By (Ax 8) we have that

$$\Gamma \vdash ((\varphi \to (\chi \to \psi)) \to ((\varphi \to \chi) \to (\varphi \to \psi)))$$

and two applications of modus ponens give $\Gamma \vdash (\varphi \to \psi)$, as required. □

2.4.2 The Completeness Theorem

The completeness theorem is one of the most fundamental results in logic. It was proven for first order logic (the big brother of propositional logic) by Kurt Gödel in his PhD thesis in 1929. It has several formulations. The most common is that everything that follows logically from a set of sentences, has a derivation from these set of sentences. A second equivalent formulation of this result is that every consistent set of formulas is satisfiable. Before we show that these two formulations are equivalent we have to show some additional facts about consistent sets.

Lemma 2.4.13. *Let Γ be a set of formulas and let φ be a formula.*

1. *If Γ is consistent, and $\Gamma \vdash \varphi$, then $\Gamma \cup \{\varphi\}$ is consistent.*

2. *If* $\Gamma \nvdash \varphi$*, then* $\Gamma \cup \{\neg\varphi\}$ *is consistent.*

Proof. We first proof (1). Suppose Γ is consistent and $\Gamma \vdash \varphi$. Assume towards a contradiction that $\Gamma \cup \{\varphi\}$ is inconsistent. Let ψ be an arbitrary formula, then $\Gamma \cup \{\varphi\} \vdash \psi$. By the deduction theorem we get that $\Gamma \vdash (\varphi \to \psi)$. Since $\Gamma \vdash \varphi$ we can use modus ponens to get $\Gamma \vdash \psi$. As ψ was arbitrary, this shows that Γ is inconsistent, a contradiction.

For (2), suppose that $\Gamma \nvdash \varphi$ and assume that $\Gamma \cup \{\neg\varphi\}$ is inconsistent. Then we get that $\Gamma \cup \{\neg\varphi\} \vdash \varphi$. By the deduction theorem, $\Gamma \vdash (\neg\varphi \to \varphi)$. We can now get a deduction of φ as follows.

$\Gamma \vdash \varphi$

1.	$(\neg\varphi \to \varphi)$	$\Gamma \vdash (\neg\varphi \to \varphi)$
2.	$((\neg\varphi \to \varphi) \to ((\neg\varphi \to \neg\varphi) \to \neg\neg\varphi))$	(Ax 9)
3.	$((\neg\varphi \to \neg\varphi) \to \neg\neg\varphi)$	1, 2 MP
4.	$(\neg\varphi \to \neg\varphi)$	Assignment
5.	$\neg\neg\varphi$	3, 4 MP
6.	$(\neg\neg\varphi \to \varphi)$	(Ax 14)
7.	φ	5, 6 MP

But this contradicts our assumption that $\Gamma \nvdash \varphi$. □

Theorem 2.4.14. *The following are equivalent:*

1. *(First formulation of completeness) For all* Γ *and all* φ*, if* $\Gamma \models \varphi$ *then* $\Gamma \vdash \varphi$*.*

2. *(Second formulation of completeness) For all* Γ*, if* Γ *is consistent, then* Γ *is satisfiable.*

Proof. We first show that (1) implies (2). Assume towards a contradiction that Γ is consistent but unsatisfiable. Let φ be such that $\Gamma \nvdash \varphi$ (such φ exists by consistency of Γ) but then as Γ is unsatisfiable, $\Gamma \models \varphi$, contradicting that (1) is true.

It remains to show that (2) implies (1). Assume that Γ is consistent implies that it is satisfiable. Suppose towards a contradiction that $\Gamma \models \varphi$ but $\Gamma \nvdash \varphi$. Then $\Gamma \cup \{\neg\varphi\}$ is consistent by Lemma 2.4.13 and therefore by

our assumption $\Gamma \cup \{\neg\varphi\}$ is satisfiable. But then $\Gamma \not\models \varphi$, a contradiction. □

We will quickly outline the idea of the proof of the completeness theorem. Looking at the implication "If Γ is consistent, then Γ is satisfiable" it is hard to see how would go about to prove this implication given some consistent set Γ. We have to construct a truth assignment e satisfying Γ.

The idea is as follows. If Γ were a set of propositional variables, then we can just define an assignment that sets all of them to true. If Γ contains negations of propositional variables. Then we know from consistency that it can not contain the propositional variables themselves, so we can set those to false in the assignment we are building.

But what if Γ contains more complex formulas, e.g., of the form $\varphi\psi$? Then we know that our truth assignment must valuate φ to true and ψ to true. So, like this we already see that there might be a way to do it by induction. But if Γ does not contain φ or ψ then we would not know from what we have done before what to do with φ and ψ. To overcome this hurdle we will do the following. Given Γ we will obtain a new set Γ^* by adding propositional variables or their negation to it so that (a) Γ^* remains consistent and (b) for every variable P, either $P \in \Gamma^*$ or $\neg P \in \Gamma^*$. We then by induction add all the possible formulas consistent with Γ^*. We end up with what is called a maximal consistent set.

Definition 2.4.7. Let Γ be a set of formulas. We say Γ is a *maximal consistent set* of formulas if Γ is consistent and if for every formula φ, either $\varphi \in \Gamma$ or $\neg\varphi \in \Gamma$.

The following proposition shows that the definition above captures the idea we described. We leave its proof as an exercise.

Proposition 2.4.15. *The set Γ is maximal consistent iff for any formula $\varphi \notin \Gamma$, $\Gamma \cup \{\varphi\}$ is inconsistent.*

Lemma 2.4.16. *Let Γ be a consistent set of formulas. Then there exists a maximal consistent set of formulas Θ such that $\Gamma \subseteq \Theta$.*

Proof. Notice that we can build an infinite list of all propositional formulas $\varphi_1, \varphi_2, \ldots$. We will now define, for each n a set of formulas Θ_n as follows.

$$\Theta_0 = \Gamma$$
$$\text{for } n \geq 1, \text{ let } \Theta_n = \begin{cases} \Theta_{n-1} \cup \{\varphi_n\} & \text{if } \Theta_{n-1} \vdash \varphi_n \\ \Theta_{n-1} \cup \{\neg\varphi_n\} & \text{otherwise} \end{cases}$$

We let $\Theta = \bigcup_{n\in\mathbb{N}} \Theta_n$ and claim that it is the desired set.

Clearly, $\Gamma = \Theta_0 \subseteq \Theta$. If φ is a formula, then $\varphi = \varphi_n$ for some n, so either $\varphi \in \Theta_n \subseteq \Theta$ or $\neg\varphi \in \Theta_n \subseteq \Theta$. It remains to show that Θ is consistent. We first show that each Θ_n is consistent.

We have that $\Theta_0 = \Gamma$ is consistent by assumption. Assume that Θ_{n-1} is consistent. If $\Theta_{n-1} \vdash \varphi_n$, then $\Theta_n = \Theta_{n-1} \cup \{\varphi_n\}$ is consistent by (1) of Lemma 2.4.13. If $\Theta_{n-1} \nvdash \varphi_n$, then $\Theta_n = \Theta_{n-1} \cup \{\neg\varphi_n\}$ is consistent by (2) of Lemma 2.4.13.

Now, assume towards a contradiction that Θ is inconsistent. So there is a formula φ such that $\Theta \vdash \varphi$ and $\Theta \vdash \neg\varphi$. Let $\psi_1, \ldots, \psi_k$ and $\gamma_1, \ldots, \gamma_l$ be derivations of φ, respectively, $\neg\varphi$ from Θ. Then there exists n such that all the formulas $\psi_1, \ldots, \psi_{k-1}$ and $\gamma_1, \ldots, \gamma_{l-1}$ are in Θ_n. But then the derivations are derivations from Θ_n and thus $\Theta_n \vdash \varphi$ and $\Theta_n \vdash \neg\varphi$ and thus Θ_n is inconsistent, a contradiction. □

We are now ready to prove the completeness theorem.

Theorem 2.4.17 (Completeness)**.** *For all Γ and all φ, if $\Gamma \models \varphi$ then $\Gamma \vdash \varphi$.*

Proof. We will prove the second formulation of the completeness theorem: "For all Γ, if Γ is consistent, then Γ is satisfiable.". This is equivalent to the statement of our theorem by Theorem 2.4.14.

Given Γ we consider a maximal consistent set $\Theta \supseteq \Gamma$. Clearly, if we show that Θ is satisfiable, then so is Γ. As Θ is maximal consistent we have that for every propositional variable P, either $P \in \Theta$, or $\neg P \in \Theta$.

Define a truth assignment e by

$$e(P) = \begin{cases} \mathbf{T} & P \in \Theta \\ \mathbf{F} & \neg P \in \Theta \end{cases}$$

We have to show that e satisfies every formula in Θ. We do this by induction by showing that $e(\varphi) = \mathbf{T}$ if and only if $\varphi \in \Theta$. The base case where P is a propositional variable is obviously true. In our induction step we will only consider the cases where $\varphi = (\chi \to \psi) \in \Theta$ and where $\varphi = \neg\psi \in \Theta$ as $\{\neg, \to\}$ is an adequate set of connectives. So assume that $\varphi = \neg\psi \in \Theta$ and that $\psi \in \Theta$ iff $e(\varphi) = \mathbf{T}$.

$$\begin{aligned} \text{Then } e(\varphi) = \mathbf{T} &\text{ iff } e(\psi) = \mathbf{F} \\ &\text{iff } \psi \notin \Theta && \text{(induction hypothesis)} \\ &\text{iff } \neg\psi \in \Theta && \text{(maximal consistency of } \Theta) \end{aligned}$$

Now, assume that $\varphi = (\chi \to \psi)$ and that for both χ and ψ, $\chi \in \Theta$ iff $e(\chi) = \mathbf{T}$ and $\psi \in \Theta$ iff $e(\psi) = \mathbf{T}$.

$$\begin{aligned} \text{Then } e(\varphi) = \mathbf{F} &\text{ iff } e(\chi) = \mathbf{T} \text{ and } e(\psi) = \mathbf{F} \\ &\text{iff } \chi \in \Theta \text{ and } \psi \notin \Theta && \text{(induction hypothesis)} \\ &\text{iff } \chi \in \Theta \text{ and } \neg\psi \in \Theta && \text{(maximal consistency of } \Theta) \\ &\text{iff } \neg(\chi \to \psi) \in \Theta && \text{(maximal consistency of } \Theta) \\ &\text{iff } (\chi \to \psi) \notin \Theta && \text{(consistency of } \Theta) \end{aligned}$$

So for all formulas φ, $e(\varphi) = \mathbf{T}$ if and only if $\varphi \in \Theta$. This implies, that e satisfies Θ. But then since $\Gamma \subseteq \Theta$, e also satisfies Γ and thus Γ is satisfiable. □

2.5 Meta vs. Object language

Notice how the symbols $\models$, $\equiv$, or $\vdash$ are not part of the language of propositional logic. Indeed, the statement $\{\varphi\} \models \psi$ is not a propositional formula.

But it is a statement about propositional logic; it says something about the relationship of propositional formulas. Such a statement is said to be a statement in the *metalanguage*. The language a statement in the metalanguage talks about is commonly called the *object language*. So, in our case the object language is propositional logic. Notice how all our definitions of the semantics of propositional logic have been done in the metalanguage. It involved the use of truth assignments, which are nothing more than functions. But in propositional logic we can not talk about functions. Here, we will not formally define what the metalanguage actually is. Defining a formal language which would allow us to prove all the theorems we are proving about propositional logic (and in the second chapter about first order logic) is possible but goes beyond the scope of this course.

The difference between metalanguage and object language becomes even more apparent now that we have a proof system and the notion of a theorem in this system. A theorem about the proof system, like the completness theorem (Theorem 2.4.17), soundness (Theorem 2.4.11), or the deduction theorem (Theorem 2.4.12) are what we usually call *metatheorems*. They talk about the system and usually one can not formulate metatheorems so that they become theorems of the system. For instance, finding a formula in propositional logic which expresses the completeness theorem is an impossible task, how would one even express the statement?

We chose to study our axiomatic derivation system, or Hilbert style system, because the proves of its metatheoremsd are quite "easy". The drawback is that it is not easy to produce a proof of a theorem in the proof system itself. There are other systems which are more intuitive to use. However, one usually sacrifices the "easy" proofs of metatheorems about them.

One needs to take special care with how symbols are used. For instance these common symbols

$$= \quad \equiv \quad \models \quad \Rightarrow \quad \Leftrightarrow \quad \vdash$$

are not symbols of propositional logic and therefore there is a difference between $P \Leftrightarrow Q$ and $\models (P \leftrightarrow Q)$.

2.6 Limits of propositional logic

Propositional logic is convenient since we have algorithms that decide whether a set of formulas is unsatisfiable and complete proof systems which help us to establish entailment. While the Hilbert style calculus which we presented does not suit itself well to be "mechanized", i.e., implemented on a computer, there are other proof systems for propositional logic which were developed with this in mind. Also, the metatheory about propositional logic was reasonably easy to develop.

However, the huge disadvantage of propositional logic is that it is simply not expressive enough to capture language and especially mathematical language as we would desire it.

Consider the following argument as an example:

> All men are mortal.
> Sokrates is a man.
> Therefore, Sokrates is mortal.

How can we formalize the first and second sentence in propositonal logic such that we can formally conclude that Sokrates is mortal? It seems impossible to establish a connection between the fact that Sokrates is mortal and that all men are mortal.

Similar examples can be found in mathematics. For instance consider the basic statement that two functions f and g agree on their inputs. A mathematician would write the "formal" statement:

$$\forall x \; f(x) = g(x)$$

How do we say this in propositional logic? How can we relate f and g? We need a stronger language to do this. We will therefore now develop the language of first order logic. Indeed within first order logic most of mathematics can be formalized.

3 First order logic

3.1 Syntax of first order logic

In propositional logic we had propositional variables as the building blocks of our formulas. For first order logic the situation is more complicated. We want to assign truth values to formulas. So what should be a formula. Should a statement like $f(x)$ have a truth value? Probably not, after all, it does not really make sense to say that $f(x)$ is true or false (unless it maps x to true or false of course).

We will now develop the syntax of first order logic with the goal of developing the notion of a formula which we can assign truth values to. Formulas in first order logic are always in a specified language.

Definition 3.1.1. A *first order language* consists of the following:

1. For every $n \geq 1$, a (possibly empty) set of *n-ary relation symbols*,
2. for every $n \geq 1$, a (possibly empty) set of *n-ary function symbols*,
3. and a (possibly empty) set of constants.

For a language $\mathcal{L}$ we often write it as $\mathcal{L} = (R_1/a_{R_1}, \ldots, f_1/a_{f_1}, \ldots, c_1, \ldots)$ where a_X is the arity of the (function) relation symbol X.

Remark. Throughout this course we assume that in every language we have a binary relation $=$.

Let us consider some examples.

Example 3.1.1. *The language $\mathcal{L}_{\leq}$ of linear orders consists of one relation symbol $\leq /2$.*

Example 3.1.2. *The language of arithmetic* $\mathcal{L}_{Arith} = \{\underline{+}/2, \underline{\cdot}/2, \underline{\leq}/2, \underline{0}, \underline{1}\}$ *where* $\underline{+}$ *and* $\underline{\cdot}$ *are function symbols,* $\underline{\leq}$ *is a relation symbol and* $\underline{0}$ *and* $\underline{1}$ *are constants.*

Example 3.1.3. *Let* $\mathcal{L} = \{H/1, M/1, s\}$ *be the language with constant* s *which we stands for "socrates", and relation symbols* H *standing for someone being human and* M *standing for someone being mortal.*

Definition 3.1.2. The *symbols* of first order logic in the language $\mathcal{L}$ are

- elements from $\mathcal{L}$,
- variables $x_1, \ldots, y_1, \ldots, z_1, \ldots,$
- logical connectives $\neg, \wedge, \vee, \rightarrow,$
- brackets (,),
- quantifiers $\exists$, $\forall$.

We call the symbols from items 2-5 logical symbols and the symbols from 1, symbols from the language.

Remark. For every language $\mathcal{L}$ we assume that (a) the sets of relation symbols, function symbols, and constants of $\mathcal{L}$ are disjoint, (b) that they do not contain variable symbols, logical connectives, brackets, or quantifiers, and (c) that they are all symbols, so their only proper prefix is the empty string.

We still want a formula to evaluate either to true **T** or **F**, but while for instance $f(x)$ has a value in some interpretation, it should not have a truth value. So while in propositional logic, all the symbols could be interpreted to have some logical meaning, in first order logic function symbols, variables and constants may occur in formulas but by themselves do not have logical meaning. They are used to build terms which in turn are the arguments for our relation symbols.

Definition 3.1.3. For a first order language $\mathcal{L}$, the *terms of* $\mathcal{L}$ are defined as follows:

1. All constants of $\mathcal{L}$ are terms of $\mathcal{L}$,
2. all variables are terms of $\mathcal{L}$,
3. if f is an n-ary function symbol of $\mathcal{L}$, and if $t_1, \ldots, t_n$ are terms of $\mathcal{L}$, then $f(t_1, \ldots, t_n)$ is a term of $\mathcal{L}$.

Example 3.1.4. *The following are terms in the language of arithmetic:*

- $\underline{+}(x, \underline{1})$
- $\underline{\cdot}(\underline{+}(x, y), \underline{1})$

The terms in the example are written in what we call *prefix notation*. For binary relations and functions we may sometimes abuse notation and write them in *infix notation*. E.g., while $x \underline{+} \underline{1}$ is formally not a term, we will still write this in our formulas and just interpret them accordingly.

We are now ready to define formulas.

Definition 3.1.4. The following are formulas of $\mathcal{L}$:

1. If t_1 and t_2 are terms of $\mathcal{L}$, then $t_1 = t_2$ is a formula of $\mathcal{L}$.
2. If R is an n-ary relation symbol of $\mathcal{L}$ and $t_1, \ldots, t_n$ are terms of $\mathcal{L}$, then $R(t_1, \ldots, t_n)$ is a formula of $\mathcal{L}$.
3. If φ_1, φ_2 are formulas of $\mathcal{L}$, then $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \to \varphi_2)$, $\neg\varphi_1$ are formulas of $\mathcal{L}$.
4. If φ is a formula and x is a variable, then $\forall x\, \varphi$ and $\exists x\, \varphi$ is a formula of $\mathcal{L}$.

Formulas satisfying (1) or (2) are called *atomic formulas*.

Example 3.1.5. *The following are formulas in $\mathcal{L}_{arith}$:*

- $\forall x(x = \underline{0} \vee \exists y\ x = y \underline{+} \underline{1})$
- $\forall y, z(y \cdot z = x \to (y = \underline{1} \vee z = \underline{1}))$

Proof. Let us quickly verify that $\forall x(x = \underline{0} \vee \exists y\ x = y \underline{+} \underline{1})$ is indeed a formula.

1.	$x = \underline{0}$	is a formula by (1)
2.	$x = y \underline{+} \underline{1}$	iaf by (1)
3.	$\exists y\ x = y \underline{+} \underline{1}$	iaf by 2 and (4)
4.	$(x = \underline{0} \vee \exists y\ x = y \underline{+} \underline{1})$	iaf by 3 and (3)
5.	$\forall x(x = \underline{0} \vee \exists y\ x = y \underline{+} \underline{1})$	iaf by 4 and (4)

□

Exercise 3.1.1. Thinking about the natural numbers together with $\underline{+}$, $\underline{\cdot}$, $\underline{0}$, and $\underline{1}$ interpreted as usual, what do the formulas say?

3.1.1 Subformulas, bound and free variables

Definition 3.1.5. Let $\mathcal{L}$ be a first order language. We define a function *sub* that given an $\mathcal{L}$-formula φ, outputs the set of all formulas it was built up from as follows:

1. If $\varphi = t_1 = t_2$ for terms t_1, t_2, then $sub(\varphi) = \{\varphi\}$,
2. if $\varphi = R(t_1, \ldots, t_n)$ for R a relation symbol in $\mathcal{L}$ and t_1,..., t_n terms of $\mathcal{L}$, then $sub(\varphi) = \{\varphi\}$.
3. if $\varphi = \neg\psi$ for a formula ψ of $\mathcal{L}$, then $sub(\varphi) = \{\varphi\} \cup sub(\psi)$.
4. if $\varphi = (\psi \circ \theta)$ where $\circ \in \{\wedge, \vee, \rightarrow\}$, then $sub(\varphi) = \{\varphi\} \cup sub(\psi) \cup sub(\theta)$,
5. if $\varphi = Qx\psi$ where $Q \in \{\forall, \exists\}$ and x a variable, then $sub(\varphi) = \{\varphi\} \cup sub(\psi)$.

A formula ψ is a *subformula* of φ if $\psi \in sub(\varphi)$.

Example 3.1.6. *Let $\mathcal{L} = \{S/1, R/3, g/1, c\}$ with S and R relations, g a function and c a constant. Calculate $sub(\varphi)$ for $(\forall x_1(\neg\forall x_2 R(x_2, c, x_1) \rightarrow x_3 = g(x_1)) \wedge (S(x_3) \vee \neg S(x_1)))$.*

Solution.

$$\begin{aligned}
sub(\varphi) =&\{\varphi\} \cup sub(\forall x_1(\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1))) \\
&\cup sub((S(x_3) \vee \neg S(x_1))) \\
=&\{\varphi, \forall x_1(\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1))), (S(x_3) \vee \neg S(x_1))\} \cup \\
&sub(S(x_3)) \cup sub(\neg S(x_1)) \cup sub((\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1))) \\
=&\{\varphi, \forall x_1(\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1))), (S(x_3) \vee \neg S(x_1)), S(x_3), \\
&\neg S(x_1), S(x_1), (\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1))\} \\
&\cup sub(\neg\forall x_2 R(x_2, c, x_1)) \cup sub(x_3 = g(x_1)) \\
=&\{\varphi, \forall x_1(\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1))), (S(x_3) \vee \neg S(x_1)), S(x_3), \\
&\neg S(x_1), S(x_1), (\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1)), \neg\forall x_2 R(x_2, c, x_1)), \\
&x_3 = g(x_1)\} \cup sub(\forall x_2 R(x_2, c, x_1)) \\
=&\{\varphi, \forall x_1(\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1))), (S(x_3) \vee \neg S(x_1)), S(x_3) \\
&, \neg S(x_1), S(x_1), (\neg\forall x_2 R(x_2, c, x_1) \to x_3 = g(x_1)), \neg\forall x_2 R(x_2, c, x_1)), \\
&x_3 = g(x_1), \forall x_2 R(x_2, c, x_1), R(x_2, c, x_1)\}
\end{aligned}$$

◀

Definition 3.1.6. The set of variables in a $\mathcal{L}$-term t, $var(t)$, is defined as follows.

1. If $t = x$ for x a variable, then $var(t) = \{x\}$,
2. if $t = c$ for c a constant, then $var(c) = \emptyset$,
3. if $t = f(t_1, \ldots, t_n)$ for $t_1, \ldots, t_n$ $\mathcal{L}$-terms and f an n-ary function of $\mathcal{L}$, then $var(t) = \bigcup_{0<i\leq n} var(t_i)$.

Definition 3.1.7. The set of variables in an $\mathcal{L}$-formula φ, $var(\varphi)$, is defined as follows.

1. If $\varphi = t_1 = t_2$ for $\mathcal{L}$-terms t_1, t_2, then $var(\varphi) = var(t_1) \cup var(t_2)$,
2. if $\varphi = R(t_1, \ldots, t_n)$ for R an n-ary relation symbol of $\mathcal{L}$, then $var(\varphi) = \bigcup_{0<i\leq n} var(t_i)$,

3. if $\varphi = \neg\psi$ for ψ an $\mathcal{L}$-formulas, then $var(\varphi) = var(\psi)$,
4. if $\varphi = (\psi \circ \theta)$ for ψ, θ $\mathcal{L}$-formulas and $\circ \in \{\wedge, \vee, \rightarrow\}$, then $var(\varphi) = var(\psi) \cup var(\theta)$,
5. and if $\varphi = Qx\ \psi$ for ψ an $\mathcal{L}$-formula and $Q \in \{\forall, \exists\}$, then $var(\varphi) = var(\psi)$.

Definition 3.1.8. The set of free variables in a formula φ, $free(\varphi)$, is defined as follows.

1. If φ is atomic, then $free(\varphi) = var(\varphi)$,
2. if $\varphi = \neg\psi$ for a formula ψ, then $free(\varphi) = free(\psi)$,
3. if $\varphi = (\psi \circ \theta)$ for formulas ψ and θ and $\circ \in \{\wedge, \vee, \rightarrow\}$, then $free(\varphi) = free(\theta) \cup free(\psi)$,
4. and if $\varphi = Qx\ \psi$ for $Q \in \{\forall, \exists\}$, ψ a formula and x a variable, then $free(\varphi) = free(\psi) \setminus \{x\}$.

If $x \in free(\varphi)$ then we say that x "occurs *free* in φ". If $x \notin free(\psi)$ for some subformula ψ of φ, then we say that x "occurs *bound* in φ". If $Qx\psi \in sub(\varphi)$ for $Q \in \{\exists, \forall\}$ and $x \in free(\psi)$ then we say that x is in the *scope* of Qx and that ψ is the scope of Qx.

Example 3.1.7. *Consider the formula* $((\forall x_1 \forall x_3 (R(x_1, x_2) \rightarrow x_1 = d) \rightarrow \neg f(x_2) = x_3) \rightarrow \forall x_3 R(x_3, c)$ *in* $\mathcal{L} = (R/2, f/1, c, d)$. *Indicate the free and bound occurrences of its variables and the scope of its quantifiers.*

Solution.

$$((\forall x_1\ \underbrace{\forall x_3\ \underbrace{(R(\overset{\mathsf{b}}{x_1}, \overset{\mathsf{f}}{x_2}) \rightarrow \overset{\mathsf{b}}{x_1} = d)}_{\text{scope } \forall x_3}}_{\text{scope } \forall x_1} \rightarrow \neg f(\overset{\mathsf{f}}{x_2}) = \overset{\mathsf{f}}{x_3}) \rightarrow \forall x_3\ \underbrace{R(\overset{\mathsf{b}}{x_3}, c)}_{\text{scope } \forall x_3})$$

◀

Definition 3.1.9. If φ is a formula with $free(\varphi) = \emptyset$, then we say that φ is an $(\mathcal{L})$-sentence.

3.2 Semantics of first order logic

Semantics of first order logic are defined using structures (also called models).

Definition 3.2.1. Let $\mathcal{L}$ be a first order language. An $\mathcal{L}$-structure $\mathcal{A}$ consists of a non-empty set A, called the *universe*, or *domain* of $\mathcal{A}$, together with

1. for every n-ary relation symbol R in $\mathcal{L}$, an n-ary relation $R^{\mathcal{A}} \subseteq A^n$,
2. for every n-ary function symbol f in $\mathcal{L}$, an n-ary function $f^{\mathcal{A}} : A^n \to A$,
3. for every constant c in $\mathcal{L}$, an element $c^{\mathcal{A}} \in A$.

Example 3.2.1. *The following are examples of structures in $\mathcal{L}_{arith}$*

- *The natural numbers with usual addition, multiplication, and order, i.e.,* $(\mathbb{N}, \underline{+} = +, \underline{\cdot} = \cdot, \underline{0} = 0, \underline{1} = 1, \underline{\leq} = \text{usual order on the naturals})$,
- *the integers with usual addition, multiplication and order on them, i.e.,* $(\mathbb{Z}, \dots)$.

Notice that while both the integers with usual addition, multiplication and order and the naturals are structures in the language of arithmetic. They are not the same. For instance, the order of the integers has no least element while the order on the natural clearly does.

Example 3.2.2. *Recall the language $\mathcal{L} = \{H/1, M/1, s\}$ from Example 3.1.3. As an example of a structure consider $\mathcal{A} = (A, H^{\mathcal{A}}, M^{\mathcal{A}}, s^{\mathcal{A}})$ with $A = \{socr, aristotle, plato, tom, jerry\}$, $H^{\mathcal{A}} = \{socr, aristotle, plato\}$, $M^{\mathcal{A}} = \{socr, aristotle, plato, tom, jerry\}$ and $s^{\mathcal{A}} = plato$.*

Definition 3.2.2. Let $\mathcal{L}$ be a language and $\mathcal{A}$ be an $\mathcal{L}$-structure. A *variable assignment* for $\mathcal{A}$ is a function from the set of variables *var* to the universe of $\mathcal{A}$, i.e. $\beta : var \to A$.

Definition 3.2.3. Let $\mathcal{A}$ be an $\mathcal{L}$-structure and β be a variable assignment for $\mathcal{A}$. Then $\beta[x/a]$ is the assignment defined by

$$\beta[x/a](y) = \begin{cases} \beta(y) & y \neq x \\ a & y = x \end{cases}.$$

Definition 3.2.4. Let $\mathcal{A}$ be a $\mathcal{L}$ structure and β be a variable assignment for $\mathcal{A}$. For all terms t of $\mathcal{L}$ we define the *evaluation of t in $\mathcal{A}$ and β, $val_\beta^\mathcal{A}$* inductively as follows.

1. If $t = c$ for c a constant, then $val_\beta^\mathcal{A}(t) = c^\mathcal{A}$,
2. if $t = x$ for x a variable, then $val_\beta^\mathcal{A}(t) = \beta(x)$,
3. if $t = f(t_1, \ldots t_n)$ for $t_1, \ldots, t_n$ $\mathcal{L}$-terms and f an n-ary function symbol in $\mathcal{L}$, then $val_\beta^\mathcal{A}(t) = f^\mathcal{A}(val_\beta^\mathcal{A}(t_1), \ldots, val_\beta^\mathcal{A}(t_n))$.

Example 3.2.3. *Evaluate $x \dot{\cdot} (\underline{1} \dot{+} \underline{1})$ in $\mathcal{L}_{arith}$ under $\mathbb{N}$, $\beta(y) = 5$ for all variables y.*

Solution.

$$\begin{aligned} val_\beta^\mathcal{A}(x \dot{\cdot} (\underline{1} \dot{+} \underline{1})) &= val_\beta^\mathcal{A}(x) \cdot val_\beta^\mathcal{A}(\underline{1} \dot{+} \underline{1}) \\ &= \beta(x) \cdot (val_\beta^\mathcal{A}(\underline{1}) + val_\beta^\mathcal{A}(\underline{1})) = 5 \cdot (1 + 1) = 10 \end{aligned}$$

◀

As one can see in the above examples, evaluations let us calculate the value of terms in a given structure.

Definition 3.2.5. We define satisfaction of a formula φ in a structure $\mathcal{A}$ relative to some variable assignment β, $\mathcal{A}, \beta \models \varphi$ inductively as follows. (We write $\mathcal{A}, \beta \not\models \varphi$ to mean "not $\mathcal{A}, \beta \models \varphi$")

1. If $\varphi = (t_1 = t_2)$, then $\mathcal{A}, \beta \models \varphi$ iff $val_\beta^\mathcal{A}(t_1) = val_\beta^\mathcal{A}(t_2)$,
2. if $\varphi = R(t_1, \ldots, t_n)$ then $\mathcal{A}, \beta \models \varphi$ iff $(val_\beta^\mathcal{A}(t_1), \ldots, val_\beta^\mathcal{A}(t_n)) \in R^\mathcal{A}$,

3. if $\varphi = \neg\psi$, then $\mathcal{A}, \beta \models \varphi$ iff $\mathcal{A}, \beta \not\models \psi$,
4. if $\varphi = (\psi \wedge \theta)$, then $\mathcal{A}, \beta \models \varphi$ iff $\mathcal{A}, \beta \models \psi$ and $\mathcal{A}, \beta \models \theta$,
5. if $\varphi = (\psi \vee \theta)$, then $\mathcal{A}, \beta \models \varphi$ iff $\mathcal{A}, \beta \models \psi$ or $\mathcal{A}, \beta \models \theta$ (or both),
6. if $\varphi = (\psi \to \theta)$, then $\mathcal{A}, \beta \models \varphi$ iff $\mathcal{A}, \beta \not\models \psi$ or $\mathcal{A}, \beta \models \theta$,
7. if $\varphi = \forall x\psi$, then $\mathcal{A}, \beta \models \varphi$ iff for every $a \in A$, $\mathcal{A}, \beta[x/a] \models \psi$,
8. and if $\varphi = \exists x\psi$, then $\mathcal{A}, \beta \models \varphi$ iff there is an $a \in A$ such that $\mathcal{A}, \beta[x/a] \models \psi$.

Example 3.2.4. *Consider again our example of the Socrates syllogism as in Example 3.2.2. We want to check whether the formula* $\forall x(H(x) \to M(x))$ *is satisfied in* $\mathcal{A}, \beta$ *where* $\beta : x \mapsto aristotle$.

Solution. We have that $\mathcal{A}, \beta \models \forall x(H(x) \to M(x))$ if and only if for all $a \in A$, $\mathcal{A}, \beta[x/a] \models (H(x) \to M(x))$. So we have to check this for all 5 elements in the universe. We do this here only for $a = tom$, as it is similar for all of them.

$$\begin{aligned}\mathcal{A}, \beta[x/tom] \models (H(x) \to M(x)) \text{ iff } & \mathcal{A}, \beta[x/tom] \not\models H(x) \text{ or } \mathcal{A}, \beta[x/tom] \models M(x)\\ \text{iff } & val^{\mathcal{A}}_{\beta[x/tom]}(x) \notin H^{\mathcal{A}} \text{ or } val^{\mathcal{A}}_{\beta[x/tom]}(x) \in M^{\mathcal{A}}\\ \text{iff } & \beta[x/tom](x) \notin H^{\mathcal{A}} \text{ or } \beta[x/tom](x) \in M^{\mathcal{A}}\\ \text{iff } & tom \notin H^{\mathcal{A}} \text{ or } tom \in M^{\mathcal{A}}\end{aligned}$$

We have $tom \notin H^{\mathcal{A}}$ so $\mathcal{A}, \beta[x/tom] \models (H(x) \to M(x))$. A similar argument shows that $\mathcal{A}, \beta[x/a] \models (H(x) \to M(x))$ for all $a \in A \setminus \{tom\}$ so we can conclude that $\mathcal{A}, \beta \models \forall x(H(x) \to M(x))$. ◀

One can only see a fundamental problem with evaluating formulas in models. To verify an $\forall$ quantifier we have to test it for every element of the universe. This is doable for finite structures. But if the structure is infinite then we can not just evaluate whether it satisfies a formula containing a $\forall$ quantifier that easily.

To see this even clearer consider the following example.

Example 3.2.5. *Consider the formula* $\exists y\ y \underline{+} \underline{1} \leq x$ *in* $\mathcal{L}_{arith}$*. Evaluate it in* $\mathbb{N}$ *under the assignments* $\beta : x \mapsto 0$ *and* $\gamma : x \mapsto 5$*.*

Solution. Let us first check that $\mathbb{N}, \gamma \models \exists y\ y \underline{+} \underline{1} \leq x$. This is the case if and only if there exists $a \in \mathbb{N}$ such that $\mathbb{N}, \gamma[y/a] \models y \underline{+} \underline{1} \leq x$. From the outside we can see that 3 should do the job. So we evaluate it at 3.

$$\mathbb{N}, \gamma[y/3] \models y \underline{+} \underline{1} \leq x \text{ iff } (val^{\mathcal{A}}_{\gamma}[y/3](y \underline{+} \underline{1}), val^{\mathcal{A}}_{\gamma}[y/3](x)) \in \leq^{\mathbb{N}}$$

We have that $val^{\mathcal{A}}_{\gamma}[y/3](y \underline{+} \underline{1}) = 4$ and $val^{\mathcal{A}}_{\gamma}[y/3](x) = 5$ and therefore $(4, 5) \in \leq^{\mathbb{N}}$. Therefore, $\mathbb{N}, \gamma[y/a] \models y \underline{+} \underline{1} \leq x$.

For β note that we have to check whether there exists an $a \in \mathbb{N}$ such that $\mathbb{N}, \beta[y/a] \models y \underline{+} \underline{1} \leq x$ and $\beta(x) = 0$. We know that there is no such y in $\mathbb{N}$ because we know $\mathbb{N}$, but to formally prove that $\mathbb{N}, \beta \not\models \exists y\ y \underline{+} \underline{1} \leq x$ we would have to try all the possibilities and there are infinitely many, so it is infeasible. ◀

On first sight it seems pedantic that we insist that we can not verify whether $\mathbb{N}, \beta \not\models \exists y\ y \underline{+} \underline{1} \leq x$ because after all we know that it is not true because we know that the natural numbers have a least element, 0. But what if we wanted to check it not $\mathbb{N}$ but for some other structure in $\mathcal{L}_{arith}$ where we don't know whether it has a least element? Well then we would actually have to search through the whole structure to verify whether the structure satisfies the formula under some given assignment β.

Proposition 3.2.6. *For all* $\mathcal{L}$*-structures* $\mathcal{A}$ *and variable assignments* β *and all* $\mathcal{L}$*-formulas* φ *and* ψ*,*

1. $\mathcal{A}, \beta \models (\varphi \wedge \psi)$ *if and only if* $\mathcal{A}, \beta \models \neg(\neg\varphi \vee \neg\psi)$*,*
2. $\mathcal{A}, \beta \models (\varphi \to \psi)$ *if and only if* $\mathcal{A}, \beta \models (\neg\varphi \vee \psi)$*,*
3. $\mathcal{A}, \beta \models \forall x\varphi$ *if and only if* $\mathcal{A}, \beta \models \neg\exists x\neg\varphi$*.*

Proof. We prove (3) and leave (1) and (2) as exercises. We have that

$$\begin{aligned}\mathcal{A}, \beta \models \forall x\varphi \text{ iff for all } a \in A\ & \mathcal{A}, \beta[x/a] \models \varphi \\ \text{iff for all } a \in A\ & \mathcal{A}, \beta[x/a] \not\models \neg\varphi \\ \text{iff there does not exist } a \in A\ & \mathcal{A}, \beta[x/a] \models \neg\varphi \\ \text{iff } & \mathcal{A}, \beta \not\models \exists x\neg\varphi \\ \text{iff } & \mathcal{A}, \beta \models \neg\exists x\neg\varphi\end{aligned}$$

□

Exercise 3.2.1. Prove (1) and (2) of Proposition 3.2.6

Proposition 3.2.6 tells us that if we want to prove semantics properties of first order logic then it is enough to assume that our formulas are made up of the symbols $\{\neg, \vee, \exists\}$. As in propositional logic we say that $\{\neg, \vee, \exists\}$ is an adequate set of symbols. We will give a formal definition of this a little later.

We want to emphasize the special role that sentences play in first order logic. Namely, that they give us information about the structures that satisfy them. We will obtain this as a corollary of the following theorem about formulas.

Theorem 3.2.7. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure. Then for any $\mathcal{L}$-formula φ and any assignments β and $\hat{\beta}$ for $\mathcal{A}$ that agree on the free variables of φ, i.e., $\beta(x) = \hat{\beta}(x)$ for all $x \in free(\varphi)$, $\mathcal{A}, \beta \models \varphi$ iff $\mathcal{A}, \hat{\beta} \models \varphi$.*

Corollary 3.2.8. *Let φ be an $\mathcal{L}$-sentence. Then for every $\mathcal{L}$-structure $\mathcal{A}$ either for every variable assignment β $\mathcal{A}, \beta \models \varphi$ or $\mathcal{A}, \beta \models \neg\varphi$.*

Let us first prove the corollary given the theorem.

Proof. If φ is a sentence, then $free(\varphi) = \emptyset$. So, for all variable assignements β and $\hat{\beta}$, $\mathcal{A}, \beta \models \varphi$ if and only if $\mathcal{A}, \hat{\beta} \models \varphi$.

Assume that $\mathcal{A}, \beta \not\models \varphi$. We have that

$$\begin{aligned}\mathcal{A}, \beta \not\models \varphi \text{ iff } & \mathcal{A}, \beta \models \neg\varphi \\ \text{iff } & \mathcal{A}, \hat{\beta} \models \neg\varphi \text{ (by Theorem 3.2.7)}\end{aligned}$$

□

It remains to prove Theorem 3.2.7. We do this by induction on formulas. The basic building blocks of atomic formulas are terms, so first we need to show that terms on the same variables evaluate to the same values.

Lemma 3.2.9. *Let $\mathcal{A}$ be a $\mathcal{L}$-structure. For any $\mathcal{L}$-term t, if β and $\hat{\beta}$ are variable assignments on $\mathcal{A}$ that agree on all variables occuring in t, then $val^{\mathcal{A}}_{\beta}(t) = val^{\mathcal{A}}_{\hat{\beta}}(t)$.*

Proof. We prove this by induction on terms. If $t = c$, a constant symbol, then for any assignments β and $\hat{\beta}$ on $\mathcal{A}$,

$$val^{\mathcal{A}}_{\beta}(t) = val^{\mathcal{A}}_{\beta}(c) = c^{\mathcal{A}} = val^{\mathcal{A}}_{\hat{\beta}}(c) = val^{\mathcal{A}}_{\hat{\beta}}(t)$$

If $t = x$ is a variable, since β and $\hat{\beta}$ agree on all variables in t we have $\beta(x) = \hat{\beta}(x)$, so

$$val^{\mathcal{A}}_{\beta}(t) = val^{\mathcal{A}}_{\beta}(x) = \beta(x) = \hat{\beta}(x) = val^{\mathcal{A}}_{\hat{\beta}}(t)$$

For the induction step assume $t = f(t_1, \ldots, t_n)$ where f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms for which the lemma holds. I.e., our hypothesis is that $val^{\mathcal{A}}_{\hat{\beta}}(t_i) = val^{\mathcal{A}}_{\beta}(t_i)$ for all $i \leq n$. Therefore

$$\begin{aligned} val^{\mathcal{A}}_{\beta}(t) &= f^{\mathcal{A}}(val^{\mathcal{A}}_{\beta}(t_1), \ldots, val^{\mathcal{A}}_{\beta}(t_n)) \\ &= f^{\mathcal{A}}(val^{\mathcal{A}}_{\hat{\beta}}(t_1), \ldots, val^{\mathcal{A}}_{\hat{\beta}}(t_n)) = val^{\mathcal{A}}_{\hat{\beta}}(t). \end{aligned}$$

□

We are now ready to prove Theorem 3.2.7.

Proof of Theorem 3.2.7. Let $\mathcal{A}$ be an $\mathcal{L}$-structure. We prove this by induction. Say $\beta(x) = \hat{\beta}(x)$ for $x \in free(\varphi)$ where $\varphi = t_1 = t_2$. Then they agree on all variables in t_1 and t_2 and

$$\mathcal{A}, \beta \models \varphi \text{ iff } val^{\mathcal{A}}_{\beta}(t_1) = val^{\mathcal{A}}_{\beta}(t_2) = val^{\mathcal{A}}_{\hat{\beta}}(t_1) = val^{\mathcal{A}}_{\hat{\beta}}(t_2) \text{ iff } \mathcal{A}, \hat{\beta} \models \varphi.$$

Say $\beta(x) = \hat{\beta}(x)$ for $x \in free(\varphi)$ where $phi = R(t_1, \ldots, t_n)$. Then they agree on all variables in t_i for all t_i, $i \leq_n$. Thus

$$\begin{aligned}\mathcal{A}, \beta \models \varphi \text{ iff } &(val^{\mathcal{A}}_{\beta}(t_1), \ldots, val^{\mathcal{A}}_{\beta}(t_n)) \in R^{\mathcal{A}} \\ &\text{iff } (val^{\mathcal{A}}_{\hat{\beta}}(t_1), \ldots, val^{\mathcal{A}}_{\hat{\beta}}(t_n)) \in R^{\mathcal{A}} \text{ iff } \mathcal{A}, \hat{\beta} \models \varphi.\end{aligned}$$

Assume $\beta(x) = \hat{\beta}(x)$ for $x \in free(\varphi)$ where $\varphi = \neg\psi$ and that if $\beta(x) = \hat{\beta}(x)$ for $x \in free(\psi)$, then $\mathcal{A}, \beta \models \psi$ iff $\mathcal{A}, \hat{\beta} \models \psi$. Then, as $free(\psi) = free(\varphi)$,

$$\mathcal{A}, \beta \models \varphi \text{ iff } \mathcal{A}, \beta \not\models \psi \text{ iff } \mathcal{A}, \hat{\beta} \not\models \psi \text{ iff } \mathcal{A}, \hat{\beta} \models \varphi.$$

Assume $\beta(x) = \hat{\beta}(x)$ for $x \in free(\varphi)$ where $\varphi = (\psi \vee \theta)$ and that if $\beta(x) = \hat{\beta}(x)$ for $x \in free(\psi)$ (for $x \in free(\theta)$), then $\mathcal{A}, \beta \models \psi$ iff $\mathcal{A}, \hat{\beta} \models \psi$ ($\mathcal{A}, \beta \models \theta$ iff $\mathcal{A}, \hat{\beta} \models \theta$). Then, as $free(\psi) \cup free(\theta) = free(\varphi)$,

$$\begin{aligned}\mathcal{A}, \beta \models \varphi \text{ iff } &\mathcal{A}, \beta \models \psi \text{ or } \mathcal{A}, \beta \models \theta \\ &\text{iff } \mathcal{A}, \hat{\beta} \models \psi \text{ or } \mathcal{A}, \hat{\beta} \models \theta \text{ iff } \mathcal{A}, \hat{\beta} \models \varphi.\end{aligned}$$

At last assume β and $\hat{\beta}$ agree on the free variables in $\varphi = \exists x\psi$ and that the theorem holds for ψ. Recall that

$$\mathcal{A}, \beta \models \varphi \text{ iff there exists } a \in A \text{ s.t. } \mathcal{A}, \beta[x/a] \models \varphi[x/a]$$

As $free(\varphi) = free(\psi) \setminus \{x\}$ we have that for every $a \in A$ $\beta[x/a](y) = \hat{\beta}[x/a](y)$ for all $y \in free(\psi)$ (including x!). So, by hypothesis for all a

$$\mathcal{A}, \beta[x/a] \models \psi \text{ iff } \mathcal{A}, \hat{\beta}[x/a] \models \psi.$$

We thus have the following.

$$\begin{aligned}\mathcal{A}, \beta \models \varphi \text{ iff } &\text{there exists } a \in A \text{ s.t. } \mathcal{A}, \beta[x/a] \models \varphi[x/a] \\ &\text{iff there exists } a \in A \text{ s.t. } \mathcal{A}, \hat{\beta}[x/a] \models \varphi[x/a] \text{ iff } \mathcal{A}, \hat{\beta} \models \varphi\end{aligned}$$

□

Corollary 3.2.8 justifies the following definition.

Definition 3.2.6. Let φ be an $\mathcal{L}$-sentence and $\mathcal{A}$ be an $\mathcal{L}$-structure. We write $\mathcal{A} \models \varphi$ if there is a variable assignment β such that $\mathcal{A}, \beta \models \varphi$.

3.3 Semantic notions

Definition 3.3.1. Let Γ be a set of $\mathcal{L}$-formulas and let φ be an $\mathcal{L}$-formula. We write $\Gamma \models \varphi$ and say that φ *is a logical consequence of* Γ, or that Γ *entails* φ if for all $\mathcal{L}$-structures $\mathcal{A}$ and variable assignments β, if for all $\gamma \in \Gamma$ $\mathcal{A}, \beta \models \gamma$ implies that $\mathcal{A}, \beta \models \varphi$

We will simply write $\mathcal{A}, \beta \models \Gamma$ if $\mathcal{A}, \beta \models \gamma$ for all $\gamma \in \Gamma$.

Note that the symbol $\models$ is overloaded, it means both being a logical consequence and for a structure and assignment to satisfy a formula. However, there should be no confusion, as it should be clear from the context what we are talking about.

Exercise 3.3.1. Show that the socrates syllogism from Example 3.2.2 is a correct argument, i.e., $\{\forall x H(x) \to M(x), H(s)\} \models M(s)$.

Proof. Suppose $\mathcal{A}, \beta \models \{\forall x H(x) \to M(x), H(s)\}$. Since $\mathcal{A}, \beta \models H(s)$ we have that $s^{\mathcal{A}} \in H^{\mathcal{A}}$. Furthermore

$$\begin{aligned} &\mathcal{A}, \beta \models \forall x H(x) \to M(x), H(s) \\ \Rightarrow &\text{for all } a \in A \ \mathcal{A}, \beta[x/a] \models (H(x) \to M(x)) \end{aligned}$$

That is, for all $a \in A$ $\mathcal{A}, \beta[x/a] \not\models H(x)$ or $\mathcal{A}, \beta[x/a] \models M(x)$. Therefore, since $s^{\mathcal{A}} \in H^{\mathcal{A}}$ we must have $s^{\mathcal{A}} \in M^{\mathcal{A}}$. So, $\mathcal{A}, \beta \models M(s)$. □

Definition 3.3.2. Let φ be an $\mathcal{L}$-formula. We say that φ is *logically valid* if $\emptyset \models \varphi$, i.e., φ is logically valid if $\mathcal{A}, \beta \models \varphi$ for all structures and assignments $\mathcal{A}, \beta$. We will write $\models \varphi$ to say that φ is valid.

Example 3.3.1. *Let $\mathcal{L} = \{f/1, R/2, c\}$, where f is a function symbol, R a relation symbol and c a constant symbol. Then the following are logically valid:*

1. $(\forall x f(x) = c \vee \neg\forall x f(x) = c)$,
2. $(\forall x R(c, x) \to R(c, y))$,
3. $\exists x(R(x, c) \to R(x, c))$.

Proof. Let $\mathcal{A}$ be any structure and β be any assignment. ad (a).

$$\begin{aligned} &\mathcal{A}, \beta \models (\forall x f(x) = c \vee \neg\forall x f(x) = c) \\ \text{iff } &\mathcal{A}, \beta \models \forall x f(x) = c \text{ or } \mathcal{A}, \beta \models \neg\forall x f(x) = c \\ \text{iff } &\mathcal{A}, \beta \models \forall x f(x) = c \text{ or } \mathcal{A}, \beta \not\models \forall x f(x) = c \end{aligned}$$

The last statement is clearly true for any structure and assignment. Thus, $\models \forall x f(x) = c \vee \neg\forall x f(x) = c$

ad (b).

$\mathcal{A}, \beta \not\models (\forall x R(c, x) \to R(c, y))$ iff $\mathcal{A}, \beta \models \forall x R(c, x)$ and $\mathcal{A}, \beta \not\models R(c, y)$.

Assume towards a contradiction that such $\mathcal{A}, \beta$ exists. Then as $\mathcal{A}, \beta \not\models R(c, y)$, $(c^{\mathcal{A}}, \beta(y)) \notin R^{\mathcal{A}}$. But $\mathcal{A}, \beta \models \forall x R(c, x)$, so for all $a \in A$ $\mathcal{A}, \beta[x/a] \models R(c, x)$. In particular $\mathcal{A}, \beta[x/\beta(y)] \models R(c, x)$, so $(c^{\mathcal{A}}, \beta(y)) \in R^{\mathcal{A}}$, a contradiction. Therefore, $\models (\forall x R(c, x) \to R(c, y))$.

ad (c).

$\mathcal{A}, \beta \models \exists x(R(x, c) \to R(x, c))$

iff there exists $a \in A$s.t. $\mathcal{A}, \beta[x/a] \models (R(x, c) \to R(x, c))$

iff it is not the case that $\mathcal{A}, \beta[x/a] \not\models R(x, c)$ and $\mathcal{A}, \beta[x/a] \models R(x, c)$

Recall that for all $\mathcal{A}$, $A \neq \emptyset$, so let $a \in A$ be arbitrary. Then it can not be that $\mathcal{A}, \beta[x/a] \models R(x, c)$ and $\mathcal{A}, \beta[x/a] \not\models R(x, c)$. So the last line is true for any $\mathcal{A}, \beta$ and all $a \in A$. As $A \neq \emptyset$ there must in particular be one such a, so $\models \exists x(R(x, c) \to R(x, c))$. □

Definition 3.3.3. Let φ and ψ be formulas in some language $\mathcal{L}$. We say that φ and ψ are logically equivalent $\varphi \equiv \psi$ if $\{\varphi\} \models \psi$ and $\{\psi\} \models \varphi$.

Example 3.3.2. *Let $\mathcal{L}$ be any language, and φ, ψ, θ be $\mathcal{L}$-formulas. Then $((\varphi \wedge \psi) \wedge \theta) \equiv (\varphi \wedge (\psi \wedge \theta))$ and $((\varphi \vee \psi) \vee \theta) \equiv (\varphi \vee (\psi \vee \theta))$.*

Proof. Assignment. □

Definition 3.3.4. Let Γ be a set of $\mathcal{L}$-formulas. We say Γ is *satisfiable* if there exists a structure $\mathcal{A}$ and a variable assignment β such that $\mathcal{A}, \beta \models \Gamma$.

Consider example (a) from Example 3.3.1. If we look at its structure it looks like $P \vee \neg P$ which is clearly a tautology in propositional logic. Indeed if we take a propositional tautology and replace the propositional variables by $\mathcal{L}$ formulas then we get a tautology in the language $\mathcal{L}$.

Let $\mathcal{L}$ be a countable first order language and enumerate all its atomic formulas:

$$\varphi_1, \varphi_2, \dots$$

To each propositional formula γ we associate a $\mathcal{L}$-formula γ^* as follows. Let $P_1, \dots$ be an enumeration of all propositional variables. If $\gamma = P_i$ we let $\gamma^\star = \varphi_i$. Say we have a formula of the form $\neg\gamma$ and γ^* is defined. Then

$$[\neg\gamma]^* = \neg\gamma^*.$$

If we have a formula $(\gamma_1 \to \gamma_2)$ and γ_1^*, γ_2^* are defined, then

$$(\gamma_1 \to \gamma_2)^* = (\gamma_1^* \to \gamma_2^*).$$

Theorem 3.3.3. *Let φ be a tautology in propositional logic, then φ^* is a first order tautology.*

Now given a structure $\mathcal{A}$ and a variable assingment β we define the following truth assignment.

$$e_{\mathcal{A},\beta}(P_i) = \begin{cases} \mathbf{T} & \mathcal{A}, \beta \models \varphi_i \\ \mathbf{F} & \mathcal{A}, \beta \not\models \varphi_i \end{cases}$$

The theorem now follows immediately from the following lemma.

Lemma 3.3.4. *Let γ be a propositional formula. Then $e_{\mathcal{A},\beta}(\gamma) = \mathbf{T}$ if and only if $\mathcal{A}, \beta \models \gamma^*$.*

Proof. The proof is by induction on formulas. If $\gamma = P_i$ it follows by definition of $e_{\mathcal{A},\beta}$.

Say for γ $e_{\mathcal{A},\beta}(\gamma) = \mathbf{T}$ if and only if $\mathcal{A}, \beta \models \gamma^*$ and consider the formula $\neg\gamma$. Then

$$e_{\mathcal{A},\beta}(\neg\gamma) = \mathbf{T} \text{ iff } e_{\mathcal{A},\beta}(\gamma) = \mathbf{F} \text{ iff } \mathcal{A}, \beta \not\models \gamma \text{ iff } \mathcal{A}, \beta \models \neg\gamma$$

and thus the lemma holds for $\neg\gamma$. Assume that $e_{\mathcal{A},\beta}(\gamma_i) = \mathbf{T}$ if and only if $\mathcal{A}, \beta \models \gamma_i$ for $i \leq 2$ and consider the formula $(\gamma_1 \to \gamma_2)$. Then

$$\begin{aligned} e_{\mathcal{A},\beta}((\gamma_1 \to \gamma_2)) = \mathbf{T} &\text{ iff } e(\gamma_1) = \mathbf{F} \text{ or } e(\gamma_2) = \mathbf{T} \\ &\text{iff } \mathcal{A}, \beta \not\models \gamma_1 \text{ or } \mathcal{A}, \beta \models \gamma_2 \\ &\text{iff } \mathcal{A}, \beta \models (\gamma_1^* \to \gamma_2^*) = (\gamma_1 \to \gamma_2)^* \end{aligned}$$

□

3.3.1 Towards a proof system

Consider the formula $(\forall x \exists y\, y = x \to \exists y\, y = \underline{1} + \underline{1})$ in $\mathcal{L}_{arith}$. This formula is of a special form in that it contains the universal statement $\forall x \exists y y = x$ as a premise and the specific statement $\exists y y = \underline{1} + \underline{1}$ as a conclusion. That statement is clearly a tautology and we can proof this easily as follows.

$$\begin{aligned} &\mathcal{A}, \beta \not\models (\forall x \exists y\, y = x \to \exists y\, y = \underline{1} + \underline{1}) \\ \text{iff } &\mathcal{A}, \beta \models \forall x \exists y\, y = x \text{ and } \mathcal{A}, \beta \not\models \exists y\, y = \underline{1} + \underline{1} \end{aligned}$$

Now we have that $\mathcal{A}, \beta \models \forall x \exists y\, y = x$ iff for all $a \in A$ $\mathcal{A}, \beta[x/a] \models \exists y\, y = x$. Then especially for $b = val^{\mathcal{A}}_{\beta[x/a]}(\underline{1} + \underline{1})$ $\mathcal{A}, \beta[x/b] \models \exists y\, y = x$. This is clearly the case if and only if $\mathcal{A}, \beta[x/b] \models \exists y\, y = \underline{1} + \underline{1}$, but there is no x in the formula so $\mathcal{A}, \beta \models \exists y\, y = \underline{1} + \underline{1}$ and $\mathcal{A}, \beta$ satisfies the formula.

We want to prove a general result about formulas of the form as above, i.e., that formulas of the form $(\forall x\ \varphi \to \psi)$, where ψ is obtained by substituting x in φ with t, are always valid. However, we need to be careful

how we do this substitution. Lets first define our substitution and then look at the reason why we need to be careful.

Definition 3.3.5. Let $\mathcal{L}$ be a first order language. For x a variable and t an $\mathcal{L}$-term, define $s^x/_t$ for all $\mathcal{L}$-terms s by induction.

1. If $s = c$ for c a constant symbol, then $s^x/_t = s$,
2. if $s = y$ for $y \neq x$ a variable, then $s^x/_t = s$,
3. if $s = x$, then $s^x/_t = t$,
4. if $s = f(t_1, \ldots, t_n)$ for $t_1, \ldots, t_n$ $\mathcal{L}$-terms and f a function symbol, then $s^x/_t = f(t_1{}^x/_t, \ldots t_n{}^x/_t)$.

Definition 3.3.6. Let $\mathcal{L}$ be a first order language. For x a variable and t an $\mathcal{L}$-term, define $\varphi^x/_t$ for all $\mathcal{L}$-formulas φ by induction.

1. If $\varphi = t_1 = t_2$ with t_1, t_2 $\mathcal{L}$-terms, then $\varphi^x/_t = t_1{}^x/_t = t_2{}^x/_t$,
2. if $\varphi = R(t_1, \ldots, t_n)$ with $t_1, \ldots, t_n$ $\mathcal{L}$-terms and R a relation symbol, then $\varphi^x/_t = R(t_1{}^x/_t, \ldots, t_n{}^x/_t)$,
3. if $\varphi = \neg\psi$ with ψ an $\mathcal{L}$-formula, then $\varphi^x/_t = \neg\psi^x/_t$,
4. if $\varphi = (\psi \circ \theta)$ with ψ and θ $\mathcal{L}$-formulas and $\circ \in \{\wedge, \vee, \rightarrow\}$, then $\varphi^x/_t = \psi^x/_t \circ \theta^x/_t$,
5. if $\varphi = Qy\psi$ where $y \neq x$ and $Q \in \{\forall, \exists\}$, then $\varphi^x/_t = Q\, y\psi^x/_t$
6. and if $\varphi = Qx\, \psi$ where $Q \in \{\forall, \exists\}$, then $\varphi^x/_t = \varphi$.

Why do we have to be careful. Consider the formula $(\forall x\ \varphi \rightarrow \varphi^x/_{y+1})$, where $\varphi = \exists y\ x = y$. This formula is

$$(\forall x \exists y\ x = y \rightarrow \exists y\, y \pm \underline{1} = y)$$

and it is clearly not logically valid. It is not true in $\mathbb{N}$ for instance. The problem is that our substitution created a new bound occurrence of y in φ.

Definition 3.3.7. Let t be an $\mathcal{L}$-term, φ be an $\mathcal{L}$-formula, and x a variable. We say that *t is substitutable for x in φ* if no variable in t has a new bounded occurrence in $\varphi^x/_t$.

Example 3.3.5. *Let R be a binary relation symbol and f be a binary function symbol. Which of the following substitutions is suitable?*

1. $\exists x R(x,y)\ ^y/_{f(x,y)}$,
2. $(\exists x R(x,y) \to y = z)\ ^z/_{f(x,y)}$,
3. $\exists x R(x,y)\ ^x/_{f(x,y)}$,

Solution. No, Yes, Yes. ◀

Lemma 3.3.6 (Substitution lemma)**.** *Let $\mathcal{L}$ be a first order language.*

1. *For any terms s and t, any variable x, and any structure and variable assignment $\mathcal{A}, \beta$ in $\mathcal{L}$, $val^{\mathcal{A}}_{\beta}(s^x/_t) = val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(s)$.*
2. *For any formula φ, for any term t and variable x, if t is substitutable for x in φ, then for all $\mathcal{A}, \beta$ in $\mathcal{L}$, $\mathcal{A}, \beta \models \varphi^x/_t$ iff $\mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \models \varphi$.*

Proof. We first prove (1) by induction on terms.

1. If $s = c$ with c a constant symbol, then $val^{\mathcal{A}}_{\beta}(s^x/_t) = c^{\mathcal{A}} = val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(s)$.
2. If $s = x$, then $val^{\mathcal{A}}_{\beta}(s^x/_t) = val^{\mathcal{A}}_{\beta}(t) = val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(s)$.
3. If $s = y$ with $y \neq x$ a variable, then $val^{\mathcal{A}}_{\beta}(s^x/_t) = \beta y = val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(s)$.
4. If $s = f(t_1, \ldots t_n)$ with $t_1, \ldots t_n$ such that $val^{\mathcal{A}}_{\beta}(t_i{}^x/_t) = val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(t_i)$ for $i \leq n$, then

$$\begin{aligned} val^{\mathcal{A}}_{\beta}(f(t_1,\ldots,t_n)^x/_t) &= val^{\mathcal{A}}_{\beta}(f(t_1{}^x/_t,\ldots,t_n{}^x/_t)) \\ &= f^{\mathcal{A}}(val^{\mathcal{A}}_{\beta}(t_1{}^x/_t), val^{\mathcal{A}}_{\beta}(t_n{}^x/_t)) \\ &= f^{\mathcal{A}}(val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(t_1),\ldots,val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(t_n)) \\ &= val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(f(t_1,\ldots,t_n)). \end{aligned}$$

(2) is again proved by induction.

1. First base case $\varphi = t_1 = t_2$, then

$$\begin{aligned}\mathcal{A}, \beta \models t_1 = t_2\ {}^{x}/_{t} &\text{ iff } \mathcal{A}, \beta \models {t_1}^{x}/_{t} = {t_2}^{x}/_{t} \\ &\text{iff } val^{\mathcal{A}}_{\beta}({t_1}^{x}/_{t_1}) = val^{\mathcal{A}}_{\beta}({t_2}^{x}/_{t_2}) \\ &\text{iff } val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(t_1) = val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(t_2) \\ &\text{iff } \mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \models t_1 = t_2\end{aligned}$$

2. Second base case $\varphi = R(t_1, \ldots, t_n)$.

$$\begin{aligned}\mathcal{A}, \beta \models R(t_1, \ldots, t_n)^{x}/_{t} &\text{ iff } \mathcal{A}, \beta \models R({t_1}^{x}/_{t}, \ldots, {t_n}^{x}/_{t}) \\ &\text{iff } (val^{\mathcal{A}}_{\beta}({t_1}^{x}/_{t}), \ldots, val^{\mathcal{A}}_{\beta}({t_n}^{x}/_{t})) \in R^{\mathcal{A}} \\ &\text{iff } (val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(t_1), \ldots, val^{\mathcal{A}}_{\beta[x/val^{\mathcal{A}}_{\beta}(t)]}(t_n)) \in R^{\mathcal{A}} \\ &\text{iff } \mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \models R(t_1, \ldots, t_n)\end{aligned}$$

3. Assume $\varphi = \neg\psi$, the theorem holds for ψ and t is substitutable for x in φ. Note that this implies that t is substitutable for x in ψ as well as any new bound occurence in $\psi^{x}/_{t}$ would yield a new bound occurrence in $\varphi^{x}/_{t}$. So for any $\mathcal{A}, \beta$,

$$\begin{aligned}\mathcal{A}, \beta \models [\neg\psi]^{x}/_{t} &\text{ iff } \mathcal{A}, \beta \not\models \psi^{x}/_{t} \\ &\text{iff } \mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \not\models \psi \qquad \text{(hypothesis)} \\ &\text{iff } \mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \models \neg\psi\end{aligned}$$

4. Assume $\varphi = (\psi \wedge \theta)$, the theorem holds for ψ and θ and t is substitutable for x in φ. By the same reasons as in (3) t is substitutable for x in ψ and θ. So for any $\mathcal{A}, \beta$,

$$\begin{aligned}\mathcal{A}, \beta \models (\psi \wedge \theta)^{x}/_{t} &\text{ iff } \mathcal{A}, \beta \models \psi^{x}/_{t} \text{ and } \mathcal{A}, \beta \models \theta^{x}/_{t} \\ &\text{iff } \mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \models \psi \text{ and } \mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \models \theta \quad \text{(hypothesis)} \\ &\text{iff } \mathcal{A}, \beta[x/val^{\mathcal{A}}_{\beta}(t)] \models (\psi \wedge \theta)\end{aligned}$$

5. Assume $\varphi = \forall x\psi$, the theorem holds for ψ, and t is substitutable for x in φ. Suppose $\mathcal{A}, \beta$ are arbitrary, then

$$\begin{aligned}\mathcal{A}, \beta \models [\forall x\psi]^x/_t \text{ iff } & \mathcal{A}, \beta \models \forall x\varphi \\ \text{iff } & \text{ for all } a \in A,\ \mathcal{A}, \beta[^x/_a] \models \psi \\ \text{iff } & \text{ for all } a \in A,\ \mathcal{A}, \beta[^x/_{val^{\mathcal{A}}_\beta(t)}][^x/_a] \models \psi \\ \text{iff } & \mathcal{A}, \beta[^x/_{val^{\mathcal{A}}_\beta(t)}] \models \forall x\psi\end{aligned}$$

6. Assume $\varphi = \forall y\psi$, the theorem holds for ψ, $y \neq x$ and t is substitutable for x in φ. Note that t substitutable for x in $\forall y\psi$ implies that y does not occur in t and that t is substitutable for x in ψ. Suppose $\mathcal{A}, \beta$ are arbitrary, then

$$\begin{aligned}\mathcal{A}, \beta \models [\forall y\psi]^x/_t \text{ iff } & \mathcal{A}, \beta \models \forall y[\psi]^x/_t \\ \text{iff for all } & a \in A,\ \mathcal{A}, \beta[^y/_a] \models \psi^x/_t \\ \text{iff for all } & a \in A,\ \mathcal{A}, \beta[^y/_a][^x/_{val^{\mathcal{A}}_{\beta[y/a]}(t)}] \models \psi && \text{(hypothesis)} \\ \text{iff for all } & a \in A,\ \mathcal{A}, \beta[^y/_a][^x/_{val^{\mathcal{A}}_\beta(t)}] \models \psi && \text{(since } y \text{ does not occur in } t\text{)} \\ \text{iff for all } & a \in A,\ \mathcal{A}, \beta[^x/_{val^{\mathcal{A}}_\beta(t)}][^y/_a] \models \psi && \text{(since } x \neq y\text{)} \\ \text{iff } & \mathcal{A}, \beta[^x/_{val^{\mathcal{A}}_\beta(t)}] \models \forall y\psi.\end{aligned}$$

□

Theorem 3.3.7. *For any formula φ, any variable x and any term t such that t is substitutable for x in φ,*

$$\models (\forall x\varphi \to \varphi\ ^x/_t).$$

Proof. Let $\mathcal{A}$, β be an arbitrary structure and variable assignment. If $\mathcal{A}, \beta \not\models \forall x\varphi$, then $\mathcal{A}, \beta \models (\forall x\varphi \to \varphi^x/_t)$ by definition of the semantics of implication. On the other hand, if $\mathcal{A}, \beta \models \forall x\varphi$, then for all $a \in A$, $\mathcal{A}, \beta[^x/_a] \models \varphi$. So, especially, $\mathcal{A}, \beta[^x/_{val^{\mathcal{A}}_\beta(t)}] \models \varphi$. Then by the substitution lemma, $\mathcal{A}, \beta \models \varphi\ ^x/_t$. So φ is a tautology. □

Proposition 3.3.8. *Let φ be an $\mathcal{L}$-formula. Then for all x, φ is a tautology if and only if $\forall x\varphi$ is a tautology.*

Proof. Assume φ is a tautology, then for all $\mathcal{A}$ and β $\mathcal{A}, \beta \models \varphi$. So also for all $\mathcal{A}, \beta$ and $a \in A$, $\mathcal{A}, \beta[x/a] \models \varphi$. Thus, $\mathcal{A}, \beta \models \forall x\varphi$.

On the other hand assume $\forall x\varphi$ is a tautology, then for all $\mathcal{A}, \beta$, $\mathcal{A}, \beta \models \forall x\varphi$. Take arbitrary $\mathcal{A}$ and β, then for all $a \in A$ $\mathcal{A}, \beta[x/a] \models \varphi$, so especially $\mathcal{A}, \beta[x/\beta(x)] \models \varphi$. But $\beta[x/\beta(x)] = \beta$, so $\mathcal{A}, \beta \models \varphi$. □

Exercise 3.3.2. Show that if φ is a tautology, then so is $\exists x\varphi$ for any x. Show that the converse is not always true.

Proposition 3.3.9. *For two $\mathcal{L}$-formulas φ and ψ, $\varphi \equiv \psi$ if and only if $\models (\varphi \leftrightarrow \psi)$.*

Proof.

$$
\begin{aligned}
\varphi \equiv \psi \text{ iff } & \varphi \models \psi \text{ and } \psi \models \varphi \\
\text{iff } & \text{for all } \mathcal{A}, \beta \text{ if } \mathcal{A}, \beta \models \varphi, \text{ then } \mathcal{A}, \beta \models \psi \\
& \quad \text{and for all } \mathcal{A}, \beta \text{ if } \mathcal{A}, \beta \models \psi, \text{ then } \mathcal{A}, \beta \models \varphi \\
\text{iff } & \text{for all } \mathcal{A}, \beta \ \mathcal{A}, \beta \not\models \varphi \text{ or } \mathcal{A}, \beta \models \psi \\
& \quad \text{and } \mathcal{A}, \beta \not\models \psi \text{ or } \mathcal{A}, \beta \models \varphi \\
\text{iff } & \text{for all } \mathcal{A}, \beta \ \mathcal{A}, \beta \models (\varphi \to \psi) \text{ and } \mathcal{A}, \beta \models (\psi \to \varphi) \\
\text{iff } & \text{for all } \mathcal{A}, \beta \ \mathcal{A}, \beta \models (\varphi \leftrightarrow \psi) \\
\text{iff } & \models (\varphi \leftrightarrow \psi)
\end{aligned}
$$

See assignment 7 for the last equivalence. □

Recall our definition of $*$ which takes a propositional formula to a formula in some fixed language $\mathcal{L}$ such that if φ is valid, then so is φ^*.

Proposition 3.3.10. *For propositional formulas φ and ψ. If $\varphi \equiv \psi$, then $\varphi^* \equiv \psi^*$.*

Proof.

$$\begin{aligned}\varphi \equiv \psi &\Rightarrow \models (\varphi \leftrightarrow \psi)\\ &\Rightarrow \models (\varphi \leftrightarrow \psi)^*\\ &\Rightarrow \models (\varphi^* \leftrightarrow \psi^*)\\ &\Rightarrow \varphi^* \equiv \psi^*\end{aligned}$$

□

Theorem 3.3.11. *Let φ be an $\mathcal{L}$-formula and ψ be a formula of φ. Suppose $\hat{\psi}$ is such that $\hat{\psi} \equiv \psi$ and let $\hat{\varphi}$ be obtained by replacing an occurrence of ψ in φ by $\hat{\psi}$. Then $\hat{\varphi} \equiv \varphi$.*

Proof. Exercise. □

3.3.2 Prenex normal form

Definition 3.3.8. A first order formula is said to be *quantifier-free* if no quantifiers occur in the formula.

Example 3.3.12. *$\neg(R(c,x) \to f(c,d) = d)$ and $x = 0 \wedge x \geq 0$ are quantifier free, while $\exists x\, x = x$ is not.*

Definition 3.3.9. A first order formula is said to be in *prenex normal form* if it has the form $Q_1 x_1 \dots Q_n x_n \psi$ where $n \geq 0$, for $1 \leq i \leq n$ x_i is a variable and $Q_i \in \{\forall, \exists\}$, and ψ is quantifier free.

Example 3.3.13. *$\forall x \exists y \exists z (R(x,z) \to f(c) = y)$ is in prenex normal form but $\forall x \exists y (\exists z R(x,z) \to f(c) = y)$ is not in prenex normal form.*

Our goal is to prove the following theorem.

Theorem 3.3.14. *Every first order formula is logically equivalent to a formula in prenex normal form.*

We will prove this theorem for formulas in which only the connectives $\neg, \wedge$ and $\forall$ occur, as every formula is equivalent to one with only these connectives.

In order to prove this theorem we will need logical equivalences that allow us to pull out quantifiers from a formula. We gather and prove these equivalences in the next lemma.

As a notational convention if Q denotes a quantifier let

$$\overline{Q} = \begin{cases} \exists & \text{if } Q = \forall \\ \forall & \text{if } Q = \exists \end{cases}$$

Lemma 3.3.15. *For any quantifier Q and all first order formulas we have the following logical equivalences.*

1. *If $\varphi \equiv \psi$, then $\neg\varphi \equiv \neg\psi$,*
2. *if $\varphi_0 \equiv \psi_0$ and $\varphi_1 \equiv \psi_1$, then $(\varphi_0 \wedge \varphi_1) \equiv (\psi_0 \wedge \psi_1)$,*
3. *if $\varphi \equiv \psi$, then $Qx\varphi \equiv Qx\psi$,*
4. *$\neg Qx\varphi \equiv \overline{Q}x\neg\varphi$,*
5. *if $x \notin free(\varphi)$, then $(\varphi \wedge Qx\psi) \equiv Qx(\varphi \wedge \psi) \equiv (Qx\psi \wedge \varphi)$,*
6. *if $y \notin free(\varphi)$ and y is substitutable for x in φ, then $Qx\varphi \equiv Qy\varphi^x/_y$.*
7. *if $x \notin free(\varphi)$, then $(\varphi \vee Qx\psi) \equiv Qx(\varphi \vee \psi) \equiv (Qx\psi \vee \varphi)$,*
8. *if $x \notin free(\varphi)$, then $(\varphi \to Qx\psi) \equiv Qx(\varphi \to \psi)$ and $(Qx\psi \to \varphi) \equiv \overline{Q}x(\psi \to \varphi)$,*

Proof. Items (1) to (3) follow directly from Theorem 3.3.11.

For (4), if $Q = \forall$, then $\overline{Q} = \exists$ and

$$\begin{aligned} \overline{Q}x\neg\varphi = \exists x\neg\varphi &\equiv \neg\forall x\neg\neg\varphi \quad \text{(by assignment, see also Proposition 3.2.6)} \\ &\equiv \neg\forall x\varphi \qquad\qquad\qquad\qquad\qquad\qquad \text{(since } \neg\neg\varphi \equiv \varphi\text{)} \end{aligned}$$

The case where $\overline{Q} = \forall$ is analogous.

For (5), suppose $x \notin free(\varphi)$ and let $\mathcal{A}, \beta$ be arbitrary. It is easy to show that $(\varphi \wedge Qx\psi) \equiv (Qx\psi \wedge \varphi)$ (exercise!). We show that $(\varphi \wedge Qx\psi) \equiv Qx(\varphi \wedge \psi)$. Assume that $Q = \forall$, the case where $Q = \exists$ is symmetric,

$$\begin{aligned}
\mathcal{A}, \beta \models (\varphi \wedge Qx\psi) \text{ iff } & \mathcal{A}, \beta \models \varphi \text{ and for all } a \in A,\ \mathcal{A}, \beta[x/a] \models \psi \\
& \text{iff for all } a \in A,\ \mathcal{A}, \beta[x/a] \models \varphi \text{ and for all } a \in A,\ \mathcal{A}, \beta[x/a] \models \psi \\
& \quad \text{(since } x \notin free(\varphi)\text{)} \\
& \text{iff for all } a \in A,\ \mathcal{A}, \beta[x/a] \models \varphi \text{ and } \mathcal{A}, \beta[x/a] \models \psi \\
& \text{iff } \mathcal{A}, \beta \models Qx(\varphi \wedge \psi)
\end{aligned}$$

For (6), suppose $y \notin free(\varphi)$ and y is substitutable for x in φ. Let $\mathcal{A}, \beta$ be arbitrary and assume that $Q = \forall$, then

$$\begin{aligned}
\mathcal{A}, \beta \models \forall x\varphi \text{ iff } & \text{for all } a \in A\ \mathcal{A}, \beta[x/a] \models \varphi \\
& \text{iff for all } a \in A\, \mathcal{A}, \beta[y/a][x/val^{\mathcal{A}}_{\beta[y/a]}(y)] \models \varphi \\
& \text{iff for all } a \in A\, \mathcal{A}, \beta[y/a] \models \varphi^x/_y \\
& \text{iff } \mathcal{A}, \beta \models \forall y\varphi^x/_y
\end{aligned}$$

If $Q = \exists$, then $\exists x\varphi \equiv \neg\forall x\neg\varphi$ and we can use Theorem 3.3.11 to get that this is equivalent to $\neg\forall y\neg\varphi^x/_y \equiv \exists y\varphi^x/_y$.

Items (7) and (8) are left as exercises. □

Example 3.3.16. *Put* $\varphi = (\forall y\neg(y = z \to \forall yy = z) \to R(y, z))$ *into prenex normalform.*

Proof. Our first goal is to rename all variables so that every variable is bound by at most one quantifier and no variable occurs bound and free. Using (6) of Lemma 3.3.15 we know that

$$\forall yy = z \equiv \forall xx = z.$$

Using Theorem 3.3.11 we get that

$$\varphi \equiv (\forall y\neg(y = z \to \forall xx = z) \to R(y, z))$$

Now using (6) and Theorem 3.3.11 again we get

$$\varphi \equiv (\forall u \neg (u = z \to \forall x x = z) \to R(y, z))$$

We now use (4) and (8) of Lemma 3.3.15 and Theorem 3.3.11 to pull the quantifiers to the front.

$$\begin{aligned}
\varphi \equiv & \exists u(\neg(u = z \to \forall x x = z) \to R(y, z)) \\
\equiv & \exists u(\neg \forall x(u = z \to x = z) \to R(y, z)) \\
\equiv & \exists u(\exists x \neg(u = z \to x = z) \to R(y, z)) \\
\equiv & \exists u \forall x(\neg(u = z \to x = z) \to R(y, z))
\end{aligned}$$

□

Exercise 3.3.3. Put $((R(z, y) \vee \neg \exists y \forall y R(y, x)) \to y = x)$ into prenex normal form.

We are now almost ready to prove Theorem 3.3.14. The proof will again be by induction but instead of inducting on the definitions of formula. We will do induction on the number of quantifiers in a formula.

Definition 3.3.10. Let φ be an $\mathcal{L}$-formula. We let $qn(\varphi)$ be the *number of quantifiers in* φ. Formally $qn(\varphi)$ is defined as follows:

1. If $\varphi$is atomic, then $qn(\varphi) = 0$,
2. if $\varphi = \neg\psi$, then $qn(\varphi) = qn(\psi)$,
3. if $\varphi = (\psi \circ \theta)$ for $\circ \in \{\wedge, \vee, \to\}$, then $qn(\varphi) = qn(\psi) + qn(\theta)$,
4. and if $\varphi = Qx\psi$, $Q \in \{\exists, \forall\}$, then $qn(\varphi) = qn(\psi) + 1$.

We will assume without loss of generality that formulas only contain the universal quantifier $\forall$ and the connectives $\neg, \wedge$. Clearly this set of symbols is adequate.

In order to prove Theorem 3.3.14 we prove the following statement.

Lemma 3.3.17. *If φ is an $\mathcal{L}$-formula with $qn(\varphi) \leq n$ for some n, then there is an $\mathcal{L}$-formula ψ in prenex normal form with $qn(\varphi) = qn(\psi)$, $free(\varphi) = free(\psi)$ and $\varphi \equiv \psi$.*

Proof. If $n = 0$ and φ is an $\mathcal{L}$-formula with $qn(\varphi) \leq 0$, then φ is quantifier free and therefore in prenex normal form.

Assume the theorem holds for formulas with number of quantifiers less or equal to n. We show it holds for $n+1$, assume that φ is a formula with $qn(\varphi) \leq n + 1$.

1. If φ is atomic then φ is quantifier free and hence in prenex normal form.

2. Suppose $\varphi = \neg\psi$ where the result holds for ψ, then if $qn(\varphi) = 0$, then φ is in prenex normal form. If $1 \leq qn(\varphi) \leq n + 1$, then $qn(\psi) = qn(\varphi)$. Since the lemma holds for ψ, there is a formula $Qx\psi_0$ in prenex normal form with

   $$qn(\psi) = qn(Qx\psi_0), \quad free(\psi) = free(Qx\psi_0), \text{ and } \psi \equiv Qx\psi_0.$$

   By (1) of Lemma 3.3.15 $\neg\psi \equiv \neg Qx\psi_0$. By (4), $\neg Qx\psi_0 \equiv \bar{Q}x\neg\psi_0$ and

   $$qn(\neg\psi_0) = qn(\psi_0) = qn(Qx\psi_0) - 1 = qn(\psi) - 1 \leq n$$

   Now, by hypothesis, there exists a formula ψ_1 in prenex normal form with

   $$qn(\psi_1) = qn(\neg\psi_0), \quad free(\psi_1) = free(\neg\psi_0), \text{ and } \psi_1 \equiv \neg\psi_0.$$

   By (3) of Lemma 3.3.15 we have $\bar{Q}x\neg\psi_0 \equiv \bar{Q}x\psi_1$. Now, clearly $\bar{Q}x\psi_1$ is in prenex normal form and

   $$\bar{Q}x\psi_1 \equiv \bar{Q}x\neg\psi_0 \equiv \neg Qx\psi_0 \equiv \neg\psi = \varphi,$$

   $qn(\bar{Q}x\psi_1) = qn(\psi_1)+1 = qn(\psi_0)+1 = qn(Qx\psi_0) = qn(\psi) = qn(\varphi)$,

   $free(\bar{Q}x\psi_1) = free(\psi_1)\backslash\{x\} = free(\neg\psi_0)\backslash\{x\} = free(\psi_0)\backslash\{x\} = free(Qx\psi_0) = free(\psi) = free(\varphi)$

   So, $\bar{Q}x\psi_1$ is as desired.

3. Suppose $\varphi = (\psi \wedge \theta)$ where the the result holds for ψ and θ. If $qn(\varphi) = 0$, then φ is in prenex normal form. If $1 \leq qn(\varphi) \leq n+1$, then since $qn(\varphi) = qn(\psi) + qn(\theta)$, either $1 \leq qn(\psi) \leq n+1$, $1 \leq qn(\theta) \leq n+1$, or both.

   Suppose $1 \leq qn(\psi) \leq n+1$. Then since the result holds for ψ, there is a formula $Qx\psi_0$ in prenex normal form with

   $$qn(Qx\psi_0) = qn(\psi), \quad free(Qx\psi_0) = free(\psi), \quad \psi \equiv Qx\psi_0.$$

   Let y be a variable such that $y \notin free(\theta) \cup free(\psi)$ and y is substitutable for x in ψ_0. Then by (6) of Lemma 3.3.15, $Qx\psi_0 \equiv Qy\psi_0{}^x/_y$. So by (2), $\varphi \equiv (Qy\psi_0{}^x/_y \wedge \theta)$. As $y \notin free(\theta)$, by (5) $\varphi \equiv Qy(\psi_0{}^x/_y \wedge \theta)$. Furthermore

   $$qn((\psi_0{}^x/_y \wedge \theta)) = qn(\psi_0{}^x/_y) + qn(\theta) = qn(\psi_0) + qn(\theta) = qn(\psi) - 1 + qn(\theta) = qn(\varphi) - 1 \leq n$$

   So by hypothesis there is a formula χ in prenex normal form with $\chi \equiv (\psi_0{}^x/_y \wedge \theta)$, $qn(\chi) = qn((\psi_0{}^x/_y \wedge \theta))$, and $free(\chi) = free((\psi_0{}^x/_y \wedge \theta))$. By (3) of the lemma $Qy(\psi_0{}^x/_y \wedge \theta) \equiv Qy\chi$ and by the above $Qy\chi \equiv \varphi$. Clearly, $Qy\chi$ is in prenex normal form so it is as desired. The case were θ has $qn(\theta) \leq n+1$ is symmetric.

4. Suppose $\varphi = \forall x\psi$ where the result holds for ψ. If $qn(\varphi) \leq n+1$, then $qn(\psi) = qn(\varphi) - 1 \leq n$. So, by hypothesis there exists a formula ψ_0 in prenex normal form with $\psi_0 \equiv \psi$, $qn(\psi_0) = qn(\psi)$, and $free(\psi_0) = free(\psi)$. Then by (3) of Lemma 3.3.15 $\forall x\psi \equiv \forall x\psi_0$. Clearly $\forall x\psi_0$ is in prenex normal form and

   $$qn(\forall x\psi_0) = 1 + qn(\psi_0) = 1 + qn(\psi) = qn(\forall x\psi)$$

   $$free(\forall x\psi) = free(\psi) \setminus \{x\} = free(\psi_0) \setminus \{x\} = free(\forall x\psi_0)$$

□

3.4 A proof system

Our proof system is defined as in the propositional case, i.e., proofs and derivations are exactly as in Definiton 2.4.2 and Definition 2.4.3. We have to define our axioms and rules of inference.

Definition 3.4.1. The axioms of our proof system are all the axiom schemes in Definition 2.4.5 (except that φ, ψ, and χ are now first order formulas and not propositional) plus the following axioms:

$(\forall x\varphi \to \varphi^x/t)$ if t is substitutable for x in φ (Ax 15)

$(\varphi \to \forall x\varphi)$ if $x \notin free(\varphi)$ (Ax 16)

$(\varphi^x/t \to \exists\varphi)$ if t is substitutable for x in φ (Ax 17)

$(\exists x\varphi \to \varphi)$ if $x \notin free(\varphi)$ (Ax 18)

$x = x$ (Ax 19)

$(x = y \to (\varphi \to \hat{\varphi}))$ if $\hat{\varphi}$ is φ with some free occurrences of x replaced by y (Ax 20)

Example 3.4.1. *Here are some examples of axioms in the language R, f where R is a binary relation and f a unary function:*

1. $(\forall x R(x, f(y)) \to (\exists x R(x, x) \to \forall x R(x, f(y)))$ *Ax 7*
2. $(\forall x R(x, y) \to R(z, y))$ *Ax 15*
3. $(\exists x R(y, f(z)) \to R(y, f(z))$ *Ax 18*

Our only rule of inference is again modus ponens. Recall (Definition 2.4.4):

Definition 3.4.2 (Modus ponens)**.** Let φ and ψ be formulas. Then *modus ponens* is the following rule:

$$\frac{(\varphi \to \psi) \qquad \varphi}{\psi}$$

Recall the definition of a derivation, the derivability or provability operator and the notion of a theorem.

Definition 3.4.3. If Γ is a set of first order formulas then a *derivation* from Γ is a finite sequence φ_1, ..., φ_n of formulas where for each $i \leq n$ one of the following holds:

1. $\varphi_i \in \Gamma$; or
2. φ_i is an axiom; or
3. φ_i follows from some φ_j (and φ_k) with $j < i$ (and $k < i$) by a rule of inference.

Definition 3.4.4. A formula φ is *derivable* from Γ, written $\Gamma \vdash \varphi$, if there is a derivation from Γ ending in φ.

Definition 3.4.5. A formula φ is a *theorem* if there is a derivation φ from the $\emptyset$. We write $\vdash \varphi$ if φ is a theorem and $\neg\varphi$ if it is not.

Lets consider the following example of a proof in our proof system.

Example 3.4.2. *Recall our example of the Socrates syllogism, Example 3.2.2. We have that* $\{\forall x(H(x) \to M(x)), H(s)\} \vdash M(s)$.

Proof.

1.	$\forall x(H(x) \to M(x))$	hyp
2.	$(\forall x(H(x) \to M(x)) \to (H(s) \to M(s)))$	Ax 15
3.	$(H(s) \to M(s))$	1, 2 MP
4.	$H(s)$	hyp
5.	$M(s)$	3, 4 MP

□

We can now establish soundness of our proof system. The proof is very similar to that of our proof system for propositional logic.

Theorem 3.4.3 (Soundness for first order logic)**.** *For any set of $\mathcal{L}$-formulas Γ and any $\mathcal{L}$-formula φ, if $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.*

Proof. We must check that all our axioms are valid and that modus ponens preserves truth in the sense that $\{\varphi, (\varphi \to \psi)\} \models \psi$. Then the result follows from the transitivity of $\models$.

First, note that if $\mathcal{A}, \beta$ are a structure and assignment such that $\mathcal{A}, \beta \models \varphi$ and $\mathcal{A}, \beta \models (\varphi \to \psi)$, then $\mathcal{A}, \beta \models \psi$ by definition of $\to$. Thus $\{\varphi, (\varphi \to \psi)\} \models \psi$.

It remains to check that our axioms are valid. Axiom 1-14 follow from the fact that they are propositional tautologies and the transfer theorem. We sketch the proof of one of them:

Validity of Axiom 7. Suppose we are given a formula of the form $(\varphi \to (\psi \to \varphi))$ where φ and ψ are first order formulas. Then there are formulas $\hat{\varphi}$ and $\hat{\psi}$ such that $\hat{\varphi}^* = \varphi$ and $\hat{\psi}^* = \psi$. Then $(\hat{\varphi} \to (\hat{\psi} \to \hat{\varphi}))$ is a propositional tautology as we have seen in the proof of soundness of propositional logic Theorem 2.4.11. By Theorem 3.3.3 so is

$$(\hat{\varphi} \to (\hat{\psi} \to \hat{\varphi}))^* = (\hat{\varphi}^* \to (\hat{\psi}^* \to \hat{\varphi}^*)) = (\varphi \to (\psi \to \varphi)).$$

That axiom 15, and axiom 17 are valid follows from Theorem 3.3.7 and the assignment.

Validity of Axiom 16 & 18. We show Axiom 16, Axiom 18 is the contraposition of Axiom 18 with $\neg\varphi$ in place of φ or, alternatively one can obtain Axiom 18 by applying Lemma 3.3.15 twice. Let $\mathcal{A}, \beta$ be an arbitrary structure and assignment. Then

$$\mathcal{A}, \beta \models (\varphi \to \forall x\varphi) \text{ iff } \mathcal{A}, \beta \not\models \varphi \text{ or for all } a \in A\ \mathcal{A}, \beta[{}^{x}/_{a}] \models \forall x\varphi$$

If $\mathcal{A}, \beta \not\models \varphi$, then the axiom is satisfied. On the other hand, if $\mathcal{A}, \beta \models \varphi$, then as $x \notin free(\varphi)$, $\mathcal{A}, \beta[{}^{x}/_{a}] \models \varphi$ for all $a \in A$, so $\mathcal{A}, \beta \models \forall x\varphi$ and thus the Axiom is satisfied.

Validity of Axiom 19. Immediate by definition of semantics.

Validity of Axiom 20. Fix arbitrary $\mathcal{A}, \beta$ such that $\mathcal{A}, \beta \models x = y$ (in the other case the axiom is satisfied anyway). This implies that $\beta(x) = \beta(y)$ and an easy induction shows that for all t, $val^{\mathcal{A}}_{\beta}(t) = val^{\mathcal{A}}_{\beta}(\hat{t})$ where $\hat{t}$ is obtained from t by replacing some occcurences of x with y. Induction over formulas then shows that the Axiom holds for any formula $\mathcal{A}, \beta \models \varphi$ if and only if $\mathcal{A}, \beta \models \hat{\varphi}$ for any formula φ ($\hat{\varphi}$ is obtained as in the axiom by replacing some free occurences of x by y). This shows the axiom is valid. □

The following definitions and theorems are all analogues of the theorems for propositional logic. Mostly their proves are very similar. We define consistent sets of formulas as for propositional logic.

Definition 3.4.6. A set Γ of $\mathcal{L}$-formulas is *consistent* if and only if there exists φ such that $\Gamma \nvdash \varphi$; it is inconsistent otherwise.

Soundness of our proof system tells us that satisfiable formulas are consistent.

Theorem 3.4.4. *If Γ is satisfiable, then it is consistent.*

Proof. Assume Γ is satisfiable. Then there exists $\mathcal{A}, \beta$ such that $\mathcal{A}, \beta \models \Gamma$. Let φ be a formula such that $\mathcal{A}, \beta \models \varphi$, then $\mathcal{A}, \beta \not\models \neg\varphi$. Now if Γ where inconsistent, then $\Gamma \vdash \neg\varphi$ and therefore also $\Gamma \models \neg\varphi$. But this implies that $\mathcal{A}, \beta \models \neg\varphi$, a contradiction. □

Notice that our system includes the same axiom schemes as the one for propositional lgoic and also only modus ponens as a rule. So, if we had a general derivation in our system for propositional logic we also have one in the system we are working in now. Thus, many theorems just trivially carry over. In particular we have theorems such as Reflexitivity, Monotonicity, and Transitivity which just follow from the concept of derivations. But also more complicated theorem such as the following follow by exactly the same proof.

Theorem 3.4.5 (Deduction Theorem)**.** *Let Γ be a set of $\mathcal{L}$-formulas and φ, ψ be $\mathcal{L}$-formulas. Then $\Gamma \vdash (\varphi \to \psi)$ if and only if $\Gamma \cup \{\varphi\} \vdash \psi$.*

Proof. Exactly the same as for Theorem 2.4.12. □

Theorem 3.4.6. *A set Γ is inconsistent if and only if there is a formula such that $\Gamma \vdash \varphi$ and $\Gamma \vdash \neg\varphi$.*

Theorem 3.4.7. *Let Γ be a set of $\mathcal{L}$-formulas. Then*

1. *If Γ is consistent and $\Gamma \vdash \varphi$, then $\Gamma \cup \{\varphi\}$ is consistent.*
2. *If $\Gamma \nvdash \varphi$, then $\Gamma \cup \{\neg\varphi\}$ is consistent.*

3.4.1 The completeness theorem

The idea of the proof the completeness theorem is similar to the idea of the proof for propositional logic. However, it is quite more complicated. We are not gonna give the full prove but instead just give a rough sketch.

Theorem 3.4.8 (Completeness)**.** *Let Γ be a set of $\mathcal{L}$-formulas and φ be an $\mathcal{L}$-formula. Then $\Gamma \models \varphi$ implies $\Gamma \vdash \varphi$.*

Instead of showing the statement above we will show the equivalent statement that "For all Γ, if Γ is consistent then Γ is satisfiable." (see Theorem 2.4.14).

Now recall how the proof of the theorem for propositional logic went. Given consistent Γ, we were constructing a truth assignment e that witnesses that Γ is satisfiable. We were doing this not by using Γ directly but by using a maximal consistent set $\Theta \supseteq \Gamma$. I.e., if $\varphi \notin \Theta$, then $\Theta \cup \{\varphi\}$ is inconsistent.

We will do something similar but this time we have to construct a structure and a variable assignment. Given $\mathcal{L}$ and Γ, how would such a structure $\mathcal{A}$ and assignment β look like. What would we take as the universe?

A good first try is to take A as all the $\mathcal{L}$-terms, i.e. $A = \{t : t\text{is an } \mathcal{L} - \text{term}\}$. But then what if $\mathcal{L}$ contains a unary function f and a constant c and $\Gamma \vdash f(c) = x$. As $f(c)$ and x are distinct terms, our structure with universe A would not do the job. Instead we might take the equivalence classes as universe, $A = \{[t] : t \text{ is an } \mathcal{L} - \text{term}\}$ (here $[t] = \{s : \Gamma \vdash s = t,\ s \text{ an } \mathcal{L} - \text{term}\}$. We can now define

$([t_1], \ldots, [t_n]) \in R^{\mathcal{A}}$ iff $\Gamma \vdash R(t_1, \ldots, t_n)$ for n-ary relation symbols R, $[f^{\mathcal{A}}([t_1], \ldots, [t_n])] = [s]$ iff $\Gamma \vdash f(t_1, \ldots, t_n) = s$, for n-ary functions, $\beta(x) = [x]$, and $c^{\mathcal{A}} = [c]$ for constants.

There is another problem, namely the quantifiers. If $\Gamma \vdash \exists x\varphi$, then we need to have a term t such that $\mathcal{A}, \beta \models \varphi^x/t$. However if we have some set Γ that is (a) consistent, (b) maximal, and contains witnesses in the sense that if $\Gamma \vdash \exists x\varphi$, then $\Gamma \vdash \varphi^x/t$ for some term t, then we can build a structure $\mathcal{A}$ that does the job.

How do we do that? We know by the assignment that if c is a fresh constant symbol, then if Γ is satisfiable, so is $\Gamma \cup \{(\exists x\varphi \to \varphi^x/c)\}$. So if Γ is maximal and $\exists x\varphi \in \Gamma$, then $\Gamma \cup \{\varphi^x/c\}$ is satisfiable. So we will extend our language by adding for every $\mathcal{L}$-sentence $\exists x\varphi$ so that $\Gamma \vdash \exists x\varphi$, a new constant c. This of course blows up our language and we end up with a set of formulas $\hat{\Gamma} \supset \Gamma$. We then blow up $\hat{\Gamma}$ to a maximal consistent set Θ in the language with constant symbols and build $\mathcal{A}$ as above for it. Then $\mathcal{A}, \beta \models \Theta$ and as $\Gamma \subseteq \Theta$ also $\mathcal{A}, \beta \models \Gamma$. As Γ does not mention any of the new constant symbols we can consider the $\mathcal{L}$-reduct $\hat{\mathcal{A}}$ of $\mathcal{A}$ (defined like $\mathcal{A}$ but lacking definitions for the new constants). Then $\hat{\mathcal{A}}$ is as required, i.e., $\hat{\mathcal{A}} \models \Gamma$ and is an $\mathcal{L}$-structure.

Of course all of this needs formal proof which we are not going to give here.

3.4.2 Limits of first order logic

Recall the compactness theorem for propositional logic **??**. The same theorem holds for first order logic. Its prove carries over directly from propositional logic.

Theorem 3.4.9. *1. If $\Gamma \vdash \varphi$, then there is finite $\Gamma_0 \subseteq \Gamma$ so that $\Gamma_0 \vdash \varphi$.*

2. If every finite subset of Γ is consistent, then Γ is consistent.

Using completeness and soundness we can get the following.

Theorem 3.4.10 (Semantic compactness)**.** *If every finite subset of Γ is satisfiable, then Γ is satisfiable.*

The completeness theorem together with the compactness theorem are useful to show limitations of first order logic. For example:

Example 3.4.11. *Let $\mathcal{L} = (E/2, c, d)$ be the language containing one binary relation, also called the language of graphs, and two constants c and d. We say that an element $b \in A$ is reachable from an element $a \in A$ if there is a path, i.e., a finite sequence $g_1, \ldots, g_n \in A$ such that $aE^{\mathcal{A}}g_1$, $g_iE^{\mathcal{A}}g_{i+1}$ for $i < n$ and $g_nE^{\mathcal{A}}b$.*

There is no $\mathcal{L}$-formula φ such that $\mathcal{A}, \beta \models \varphi$ iff $d^{\mathcal{A}}$ is reachable from $c^{\mathcal{A}}$.

Proof. We can consider formulas saying that d is not reachable from c in less than n steps:

$$\psi_n = \forall x_1 \dots \forall x_n \ (\neg cEx_1 \vee \neg x_1 Ex_2 \cdots \vee \neg x_n Ed).$$

Assume φ was a formula saying that d is reachable from c and consider $\Gamma = \varphi \cup \{\psi_n : n \in \mathbb{N}\}$. Note that $\{\psi_n : n \in \mathbb{N}\}$ says that d is not reachable from c and thus Γ can not be satisfiable. Observe that every finite subset of Γ is satisfiable. So by compactness Γ is satisfiable. But this is impossible, therefore the formula φ can not express that d is reachable from c.

□

To express such a property as reachability in the language of graphs we need an even stronger logic. The downside is that logics capable of expressing this lack other nice features you would want of a logic such as and especially compactness.